Τεχνολογικό Εκπαιδευτικό Ίδρυμα Κρήτης

Σχολή Τεχνολογικών Εφαρμογών
Τμήμα Εφαρμοσμένης Πληροφορικής & Πολυμέσων

Πτυχιακή εργασία

Τίτλος: Αυτόνομο σύστημα παρακολούθησης και ελέγχου περιβαλλοντικών συνθηκών βασισμένο στην πλατφόρμα ARM mini2440

Καλαμπούκας Χρήστος Α.Μ. 1024

Επιβλέπων καθηγητής: Γεώργιος Κορνάρος

Επιτροπή Αξιολόγησης: Βιδάκης Νικόλαος, Μανιφάβας Χαράλαμπος

Ημερομηνία παρουσίασης: 23/05/20013

Ευχαριστίες

Θα ήθελα να ευχαριστήσω για τη συνεισφορά τους στην παρούσα εργασία:
Τον bill του station51.net (Bill Sargent) για τα πολύ καλά how to που δημοσιεύει στο forum του.

Τον emeb για της πολύ καλές οδηγίες του και τα αρχεία για την ενσωμάτωση SPI, GPIO υποστήριξης στον πυρήνα.

Τον busserror που (σύμφωνα με όσα διάβασα) έκανε τις τροποποιήσεις για τη mini2440 στον πυρήνα.

Τον Παύλο που έφτιαξε το καλώδιο που ενώνει την πλακέτα με τον αισθητήρα.

Την Όλγα για τη ορθογραφική και συντακτική επιμέλεια.

Περίληψη

Τίτλος: Αυτόνομο σύστημα παρακολούθησης και ελέγχου περιβαλλοντικών συνθηκών βασισμένο στην πλατφόρμα ARM mini2440.

Σκοπός της συγκεκριμένης εργασίας είναι, να αναπτυχθεί σύστημα, με σκοπό τον έλεγχο και την καταγραφή της θερμοκρασίας και της υγρασίας σε ένα χώρο. Ως υλικό χρησιμοποιήθηκε, η πλακέτα mini2440 της FriendlyARM και ο αισθητήρας DC-SS500 της Sure Electronics. Ως λογισμικό της πλακέτας και του υπολογιστή ελέγχου επισημοποιήθηκε Linux και προτιμήθηκε, όπου ήταν εφικτό ελεύθερο λογισμικό και λογισμικό ανοιχτού κώδικα.

Η εργασία, το λογισμικό και το υλικό της πλακέτας εμπίπτουν στο πεδίο των ενσωματωμένων συστημάτων, επομένως γίνεται λόγος για τα συστήματα αυτά και τις ιδιαιτερότητες τους. Ενσωματωμένο ονομάζεται ένα σύστημα που αναλαμβάνει μια πολύ συγκεκριμένη εργασία, ως μέρος ενός μεγαλύτερου συστήματος. Τέτοια συστήματα είναι συχνά ηλεκτρικές ή ηλεκτρονικές συσκευές καθημερινής χρήσης.

Το λογισμικό που αναπτύχθηκε περιλαμβάνει ένα πρόγραμμα που εκτελείται στο παρασκήνιο και μια διεπαφή σε μορφή ιστοσελίδας. Το πρόγραμμα παρασκηνίου λαμβάνει, μετρήσεις θερμοκρασίας και υγρασίας και τις αποθηκεύει. Επιπλέον, ελέγχει δύο LED, το ένα ανάβει όταν η θερμοκρασία ή η υγρασία ξεπεράσει, κάποιες μέγιστες τιμές και το άλλο όταν πέσει κάτω από κάποια ελάχιστα. Η διεπαφή δίνει στον χρήστη τη δυνατότητα, να ρυθμίσει τις μέγιστες και ελάχιστες τιμές στο πρόγραμμα παρασκηνίου, να παρακάμψει αυτές τις τιμές αναβοσβήνοντας τα LED κατά βούληση, να δει την τρέχουσα θερμοκρασία και υγρασία, να αναζητήσει παλιότερες μετρήσεις.

Το σκεπτικό είναι ότι με την αντικατάσταση των LED με συσκευές ελέγχου θερμοκρασίας και υγρασίας, όπως κάποιο κλιματιστικό, αερόθερμο ή ανεμιστήρα, θα μπορούσαμε να παρακολουθούμε τις περιβαλλοντικές συνθήκες κάποιου χώρου, να τις επηρεάζουμε χειροκίνητα ή να αφήνουμε το σύστημα να διατηρήσει, τα επιθυμητά επίπεδα θερμοκρασίας και υγρασίας ρυθμίζοντας κατάλληλα τα μέγιστα και ελάχιστα. Ο προγραμματισμός έγινε στη C, ενώ η διεπαφή γράφηκε σε C cgi ώστε να είναι προσβάσιμη ακόμα και μέσω διαδικτύου.

Abstract

Title: Automated system for the surveillance and control of environmental variables based on ARM mini2440 platform.

The objective of this project is the development of a system to monitor and store the temperature and humidity of an area. The hardware used is the board mini2440 by FriendlyARM and the sensor DC-SS500 by Sure Electronics. The software for both the board and the host PC is Linux, free software and open source software is preferred whenever possible.

This project, the board's hardware and software belong in the field of embedded systems, so some details are given about these systems and their special properties. An embedded system is designed for a specific job and is part of a larger system. Every day use electric and electronic devices are often that kind of systems.

The developed software includes a background program and a web interface. The background program measures and stores temperature and humidity values. In addition it controls two LEDs, one of which lights when temperature or humidity exceeds some maximum values, the other lights when one of them goes below some minimum values. The interface gives the ability to set the minimum and maximum values for the background program, to ignore these values and use the LEDs manually, to observe the current temperature and humidity values and to browse older measures.

The idea is that, by replacing the LEDs by temperature and humidity control devices, like an air condition, a heater or a fan, we could monitor the environmental variables of a place, affect them manually or let the system preserve the desirable temperature and humidity by setting the right minimum and maximum values. All programs are written in C. While the interface is written in C cgi making it accessible even via internet.

Περιεχόμενα

Πίνακας εικόνων

Πίνακας πινάκων

Κεφάλαιο 1 - Εισαγωγή

1.1 Εισαγωγή

Η παρούσα πτυχιακή ασχολείται με τον προγραμματισμό ενσωματωμένων συστημάτων, σε περιβάλλον Linux, τόσο στο ίδιο το ενσωματωμένο σύστημα, όσο και στον υπολογιστή ελέγχου. Στα πλαίσια αυτού αναλύεται ο τρόπος εγκατάστασης λειτουργικού σε ενσωματωμένο σύστημα, καθώς και ο τρόπος εγκατάστασης και ρύθμισης κάποιων εφαρμογών.

Τα ενσωματωμένα συστήματα είναι πολύ διαδεδομένα, και οι περισσότεροι δε γνωρίζουν ότι τα χρησιμοποιούν καθημερινά. Με την έλευση των έξυπνων τηλεφώνων και των υπολογιστών ταμπελών υπήρξε μια έκρηξη στην ανάπτυξη εξελιγμένων λειτουργικών για τέτοια συστήματα. Η εξάπλωση τους τα επόμενα χρόνια αναμένεται να είναι τεράστια λόγω της βελτίωσης των επιδόσεων τους και της μειωμένης τιμής τους. Ολοένα και περισσότερες συσκευές, που θα μπορούσαν να κατασκευαστούν με τη χρήση ολοκληρωμένων κυκλωμάτων, τελικά κατασκευάζονται ως ολοκληρωμένα συστήματα, λόγω χαμηλότερου κόστους και αυξημένης ευελιξίας.

Σε τέτοια συστήματα το Linux και γενικότερα το ελεύθερο λογισμικό, απολαμβάνουν πολύ μεγάλη αποδοχή. Το χαμηλό κόστος και οι δυνατότητες παραμετροποίησης που προσφέρει, το κάνουν ιδανική επιλογή για το χώρο των ενσωματωμένων συστημάτων. Αξίζει να αναφέρουμε ότι οι διαφορές από το ένα σύστημα στο άλλο είναι τεράστιες και το κόστος πρέπει να κρατηθεί σε όσο το δυνατό χαμηλά επίπεδα.

Κάτι σημαντικό για τη συγκεκριμένη εργασία είναι οι θύρες GPIO και το πρωτόκολλο επικοινωνίας SPI, που δίνουν τη δυνατότητα σε τέτοια συστήματα να συνδεθούν με διάφορες συσκευές και εξαρτήματα. Τέτοιες συσκευές μπορεί να είναι: αισθητήρες θερμοκρασίας, υγρασίας, οξύτητας και άλλοι, LED, διακόπτες και με λίγη περισσότερη δουλειά στην καλωδίωση, με συστήματα θέρμανσης ή εξαερισμού.

Στο τέλος με τη χρήση της αναπτυξιακής πλακέτας Mini2440 της FriendlyARM και τον αισθητήρα θερμοκρασίας και υγρασίας DC-SS500 της Sure Electronics αναπτύσσεται μια εφαρμογή καταγραφής μετρήσεων. Μια διεπαφή σε περιβάλλον ιστοσελίδας, δίνει τη δυνατότητα παρακολούθησης των μετρήσεων και ελέγχου δύο LED. Τα LED απεικονίζουν την ανάδραση που μπορεί να δοθεί σε μια συσκευή. Αν αντί για LED συνδέονταν ένας ανεμιστήρας εξαερισμού για παράδειγμα θα μπορούσε να μειώσει την υγρασία.

1.2 Κίνητρο για την Διεξαγωγή της Εργασίας

Κατά τη διάρκεια της φοίτησης μου, είχα την ευκαιρία να δουλέψω με κάποιους μικρο-ελεγκτές στα εργαστήρια της ρομποτικής και των μικροϋπολογιστών. Το γεγονός ότι, οι μικρο-ελεγκτές με τους αισθητήρες και τους επενεργητές που συχνά συνδέονται πάνω τους, αντιδρούν με τον πραγματικό κόσμο, τους δίνει ιδιαίτερο ενδιαφέρον. Οι μικρο-ελεγκτές που χρησιμοποιήσαμε στα εργαστήρια δεν είχαν λειτουργικό σύστημα και απλώς έτρεχαν το πρόγραμμα που φορτώναμε στη μνήμη τους.

Η ιδέα της εν λόγω πτυχιακής, ήταν να χρησιμοποιηθεί μια, ανώτερων προδιαγραφών, αναπτυξιακή πλακέτα, για την καταγραφή περιβαλλοντικών μεταβλητών, παρέχοντας κάποια ανάδραση. Η πλακέτα που επελέγη χρησιμοποιεί λειτουργικό σύστημα, και δίνει πολλές δυνατότητες επικοινωνίας. Στη θέση του λειτουργικού συστήματος επελέγη το Linux. Έτσι εκτός από τις γνώσεις σχετικά με τους μικρο-ελεγκτές θα αποκομίσω και καλύτερη κατανόηση του συγκεκριμένου λειτουργικού.

1.3 Σκοπός και Στόχοι Εργασίας

Σκοπός είναι να γραφεί μία εφαρμογή που θα διαβάζει τις μεταβλητές θερμοκρασίας και υγρασίας και θα τις αποθηκεύει σε μία βάση δεδομένων. Ανάλογα με τις μεταβλητές που διαβάζει μπορεί να παρέχει κάποια ανάδραση, η οποία θα περιοριστεί σε άναμμα ή σβήσιμο LED. Σε συνθήκες πρακτικής χρήσης, θα μπορούσε να ενεργοποιεί και να απενεργοποιεί αυτόματα συσκευές, όπως

θέρμανση και εξαερισμός για τη βελτίωση των συνθηκών του χώρου. Τα αποτελέσματα των μετρήσεων θα γίνονται προσβάσιμα μέσα από μια διαδικτυακή διεπαφή. Θα δίνεται επιπλέον δυνατότητα ελέγχου της ανάδρασης, καθώς και χειροκίνητο έλεγχο των LED. Καθώς η διεπαφή θα είναι προσβάσιμη μέσω φυλλομετρητή, θα μπορεί να συνδεθεί και στο διαδίκτυο δίνοντας έλεγχο της εφαρμογής από οπουδήποτε.

1.4 Δομή πτυχιακής

Το κεφάλαιο 1 είναι εισαγωγικό. Αναφέρονται σε αυτό οι σκοποί και οι στόχοι της πτυχιακής καθώς και λίγα λόγια για το θέμα της.

Στο κεφάλαιο 2 εξηγούνται κάποια πράγματα που είναι καλό να ξέρει κάποιος πριν προχωρήσει περαιτέρω.

Στο κεφάλαιο 3 αναλύονται οι τεχνολογίες και ο εξοπλισμός που χρησιμοποιήθηκαν και δίνεται μια λίστα στόχων που έπρεπε να επιτευχθούν για την υλοποίηση της εργασίας.

Το κεφάλαιο 4 αναλύει διεξοδικά το λογισμικό που χρησιμοποιήθηκε, δίνοντας όπου είναι απαραίτητο οδηγίες για τον τρόπο χρήσης του.

Στο κεφάλαιο 5 γίνεται ανάλυση των εφαρμογών που αναπτύχθηκαν. Αναλύονται οι δυνατότητες τους, καθώς και ο τρόπος λειτουργίας τους.

Το κεφάλαιο 6 αναφέρει τα συμπεράσματα στα οποία οδηγούν τα υπόλοιπα κεφάλαια. Αναλύει επιπλέον σκέψεις για μελλοντική βελτίωση ή επέκταση της εργασίας.

Κεφάλαιο 2 - Χρήσιμες πληροφορίες για τα ενσωματωμένα συστήματα

2.1 Γενικά για τα ενσωματωμένα συστήματα

Ενσωματωμένο (embedded) λέγετε ένα σύστημα βασισμένο σε μικροελεγκτή (misroprosessor), που αναλαμβάνει μια συγκεκριμένη εργασία. Πιο αναλυτικά, ένα ενσωματωμένο σύστημα:

1. Είναι φτιαγμένο να κάνει μια συγκεκριμένη δουλειά ή δουλειές.

Σε αντίθεση με τους συνηθισμένους υπολογιστές (που πολλοί αποκαλούν “υπολογιστές γενικής χρήσης”), οι ενσωματωμένοι δε μπορούν να προγραμματιστούν από το χρήστη. Ενώ οι υπολογιστές που έχει, ο μέσος άνθρωπος στο μυαλό του, μπορούν να αλλάξουν εντελώς τη λειτουργία τους (σταθμός πολυμέσων, κειμενογράφος, παιχνιδομηχανή) ανάλογα με το πρόγραμμα που θα εκτελέσει ο χρήστης. Αντίθετα τα ενσωματωμένα συστήματα έχουν ειδικού σκοπού υλικό και λογισμικό για να φέρνουν σε πέρας συγκεκριμένες εργασίες. Ο χρήστης είναι συνήθως πολύ δύσκολο να επέμβει.

2. Είναι ενσωματωμένο σε μία συσκευή.

Όπως προδίδει το όνομά τους, τα ενσωματωμένα συστήματα, είναι ενσωματωμένα σε συσκευές. Αν και λίγοι το αντιλαμβάνονται, τα χρησιμοποιούμε καθημερινά. Οι φούρνοι μικροκυμάτων, τα πλυντήρια, τα modem-router και πολλές άλλες συσκευές, μπορεί να είναι ενσωματωμένα συστήματα. Περιέχουν μικροεπεξεργαστή, μνήμη και κάποιο λογισμικό χάρη στα οποία αυτοματοποιούν τις λειτουργίες τους. Τα σύγχρονα αυτοκίνητα έχουν τέτοια συστήματα, για τον έλεγχο καυσαερίων, τα φρένα και άλλους αυτοματισμούς. Τα πολυτελή αυτοκίνητα μπορεί να έχουν πάνω από εξήντα μικροελεγκτές, ελέγχοντας με αυτούς ακόμα και τον εσωτερικό φωτισμό.

Συχνά τα ενσωματωμένα συστήματα είναι συστήματα πραγματικού χρόνου, δηλαδή συστήματα που ύψιστη προτεραιότητα είναι η εκτέλεση λειτουργιών σε αυστηρά καθορισμένο χρόνο. Δε θα αναλυθούν όμως περαιτέρω γιατί δε σχετίζονται με το περιεχόμενο αυτής της εργασίας.

Μερικές φορές είναι δύσκολο να πούμε αν μια συσκευή λειτουργεί με ενσωματωμένο σύστημα ή όχι, γιατί οι λειτουργίες που προσφέρει μπορούν να επιτευχθούν με ένα ολοκληρωμένο κύκλωμα. Βέβαια η επιλογή του ενσωματωμένου συστήματος είναι πιο ευέλικτη, ευκολότερη και πολλές φορές φθηνότερη. Εδώ πρέπει να αναφερθεί ότι ένας υπολογιστής γενικού σκοπού περιέχει πολλά ενσωματωμένα συστήματα. Ακόμα και το πληκτρολόγιο, το ποντίκι, ο σκληρός δίσκος μπορεί να είναι τέτοια συστήματα με δικό τους επεξεργαστή και λογισμικό, ειδικευμένα στις λειτουργίες που πρέπει να εκτελέσουν.

Άλλες διαφορές αφορούν τις μονάδες εισόδου εξόδου και τις θύρες επικοινωνίας. Για εξοικονόμηση πόρων και απλοποίηση της κατασκευής, αποφεύγονται πολύπλοκοι τρόποι επικοινωνίας, που έχουμε συνηθίσει στους υπολογιστές γενικής χρήσης. Αντί γι' αυτά ενσωματώνουν πιο χαμηλού επιπέδου πρωτόκολλα όπως SPI και I2C.

Βέβαια μερικές φορές η ψαλίδα ανάμεσα σε ενσωματωμένο και γενικού σκοπού σύστημα φαίνεται να μικραίνει. Καθώς οι δυνατότητες των ενσωματωμένων συστημάτων βελτιώνονται, εξοπλίζονται με πρωτόκολλα επικοινωνίας που παραδοσιακά συναντάμε σε γενικού σκοπού υπολογιστές όπως USB, Ethernet, Wifi ακόμα και HDMI. Επιπλέον, αν και δεν είναι πάντα εύκολο, είναι εφικτό να προγραμματίσουμε κάποια ενσωματωμένα συστήματα, ώστε να κάνουν διαφορετικά πράγματα από ό,τι είχε προβλέψει ο κατασκευαστής. Πλέον πολλά ενσωματωμένα τρέχουν Linux διανομές, και επομένως είναι εφικτό να εγκαταστήσουμε σε αυτά πλήθος εφαρμογών ή ακόμα και να τρέξουμε πολλές εφαρμογές ταυτόχρονα. Ειδικά μετά την εμφάνιση των έξυπνων κινητών τηλεφώνων και των υπολογιστών ταμπλέτας είναι δύσκολο να πούμε τι είναι ενσωματωμένο σύστημα και τι όχι.

2.2 Ιστορία ενσωματωμένων συστημάτων

Η όλη ιστορία ξεκίνησε από την παραγωγή ηλεκτρονικών αριθμομηχανών τη δεκαετία του 1970. Αρχικά ο σχεδιασμός τους γινόταν με λογικές πύλες. Καθώς τα ολοκληρωμένα κυκλώματα εξελίσσονταν, συχνά κομμάτια υλικού, όπως ένας αθροιστής μπορούσαν να αγοραστούν και να τοποθετηθούν στο κύκλωμα. Η μεγάλη αλλαγή έγινε το 1971. Η εταιρεία Busicom, δύο χρόνια πριν,

ζήτησε από την intel μια σειρά από διαφορετικά ολοκληρωμένα για τις νέες αριθμομηχανές της. Η intel αποφάσισε, αντί να κατασκευάσει διαφορετικού τύπου υλικό για κάθε μοντέλο αριθμομηχανής, να δημιουργήσει ένα γενικού τύπου ολοκληρωμένο που θα μπορούσε να προσαρμοστεί στις ανάγκες της κάθε συσκευής.



Έτσι προέκυψε ο 4004 της intel. Ο πρώτος γενικού σκοπού μικροεπεξεργαστής, που προσάρμοζε τη συμπεριφορά του διαβάζοντας μια εξωτερική μνήμη. Η Busicom, απλώς συνέδεε τον 4004 στο υπόλοιπο υλικό της αριθμομηχανής και πέρναγε στη μνήμη το κατάλληλο λογισμικό. Θα μπορούσαμε να πούμε ότι αυτές οι αριθμομηχανές ήταν τα πρώτα ενσωματωμένα συστήματα. Η εξάπλωση των μικροελεγκτών ήταν ταχύτατη. Οι πρώτες εφαρμογές τους περιελάμβαναν, φωτεινούς σηματοδότες, εφαρμογές στην έρευνα του διαστήματος και συστήματα αεροπορικής πλοήγησης.

2.3 ARM

Η Αρχιτεκτονική ARM περιγράφει μια οικογένεια επεξεργαστών σχεδιασμένων με βάση τις αρχές της αρχιτεκτονικής RISC (2.4 Αρχιτεκτονική RISC). Αναπτύχθηκε από τη βρετανική ARM Holdings και παλιότερα ήταν γνωστή σαν (*Acorn RISC Machine*). Η ίδια η ARM Holdings δεν κατασκευάζει επεξεργαστές, αλλά πουλάει διάφορες άδειες χρήσης σε τρίτες εταιρείες, δίνοντας τους έναν εύκολο τρόπο να παράγουν τους δικούς τους επεξεργαστές, προσαρμοσμένους στις ανάγκες τους.

Η αρχιτεκτονική χωρίζεται σε πολλές εκδόσεις και παραλλαγές. Η κάθε έκδοση έχει βελτιώσεις σε σχέση με την προηγούμενη, ενώ για την κάθε εταιρεία που προτίθεται να κατασκευάσει ARM επεξεργαστές, μπορούν να γίνουν τροποποιήσεις. Αυτή τη στιγμή η τελευταία έκδοση της αρχιτεκτονικής είναι η ARMv8.

Χάρη στην αρχιτεκτονική RISC, οι ARM επεξεργαστές είναι απλοί κατασκευαστικά. Ο βασικός πυρήνας τους, χρειάζεται για να υλοποιηθεί 35.000 τρανζίστορ, που είναι πολύ λίγα σε σύγκριση με μερικά εκατομμύρια στους περισσότερους συμβατούς επεξεργαστές. Αυτό τους κάνει πολύ οικονομικούς σε ενέργεια και επομένως ιδανικούς για μικρές συσκευές, ειδικά αν είναι φορητές.

Από τη δεκαετία του 1980 που πρωτοεμφανίστηκαν μέχρι σήμερα οι επεξεργαστές ARM έχουν γνωρίσει τεράστια αποδοχή. Αυτή τη στιγμή αποτελούν μια από τις συχνότερες επιλογές για smartphones, tablet PC, ψηφιακές τηλεοράσεις και άλλες μικρές συσκευές, ενώ υπάρχουν βλέψεις εξάπλωσης τους και σε άλλα συστήματα όπως οι υβριδικοί υπολογιστές με χαρακτηριστικά ταμπλέτας και φορητού ή ακόμα και servers που, εκμεταλλευόμενοι τη μικρή ενεργειακή κατανάλωση, θα μπορούν να ενσωματώνουν πολλούς επεξεργαστές.

2.4 Αρχιτεκτονική RISC

Η αρχιτεκτονική RISC (Reduced Instruction Set Computing) διακρίνεται για το μικρό μέγεθος εντολών της. Έρχεται σε αντίθεση με το CISC (Complex Instruction Set Computing) όπου πιο πολύπλοκες εντολές χρησιμοποιούνται. Το σκεπτικό είναι ότι αν και χρειάζονται περισσότερες εντολές RISC για να υλοποιηθεί το ίδιο πρόγραμμα (λόγο της απλότητας τους), θα εκτελεστούν πιο γρήγορα, λόγω του ελάχιστου χρόνου που απαιτεί η εκτέλεση της κάθε μιας. Γενικά ο χρόνος εκτέλεσης της κάθε εντολής RISC είναι ένας κύκλος ρολογιού.

RISC επεξεργαστές από τεχνική σκοπιά:

- Η πρόσβαση στη μνήμη γίνεται με μεμονωμένες εντολές και όχι ως μέρος εκτέλεσης μίας εντολής.
- Η κάθε εντολή έχει μέγεθος όσο μια λέξη του επεξεργαστή (32 bit για τον ARM9 της mini2440).
- Οι περισσότεροι καταχωρητές είναι γενικού τύπου, όσον αφορά το υλικό.

Γενικά όλα τα χαρακτηριστικά της RISC αρχιτεκτονικής ωθούν στην απλότητα του υλικού και των εντολών. Έτσι επιτυγχάνεται αξιοπιστία και σταθερότητα.

2.5 Ενσωματωμένα λειτουργικά συστήματα

Καθώς τα ενσωματωμένα συστήματα είναι ουσιαστικά ηλεκτρονικοί υπολογιστές, χρειάζονται και κάποιο λειτουργικό σύστημα. Λόγω των διαφορών τους, με τους, γενικής χρήσης, υπολογιστές, το λειτουργικό αυτό είναι συνήθως πολύ διαφορετικό και λόγω των μεγάλων διαφορών που έχουν τα ενσωματωμένα συστήματα μεταξύ τους, τα ενσωματωμένα λειτουργικά διαφέρουν πολύ το ένα από το άλλο.

Σε γενικές γραμμές τα ενσωματωμένα λειτουργικά συστήματα πρέπει:

- Να είναι μικρά σε μέγεθος, γιατί ο αποθηκευτικός χώρος είναι περιορισμένος
- Να χρησιμοποιούν με φειδώ τους πόρους του συστήματος, καθώς είναι λίγοι
- Να είναι αξιόπιστα γιατί οποιαδήποτε πρόβλημα είναι δύσκολο έως αδύνατο να διορθωθεί από το χρήστη

Από την άλλη, δε χρειάζεται να έχουν πολλά από τα χαρακτηριστικά που θεωρούνται απαραίτητα σε ένα γενικής χρήσης σύστημα, αφού το εύρος εργασιών τους είναι συγκεκριμένο.

Στην πιο απλή μορφή, το λειτουργικό είναι ένα με την εφαρμογή. Ένα μόνο αρχείο ενσωματώνει τις απαραίτητες λειτουργίες του λειτουργικού και της εφαρμογής, χωρίς να μπορεί να φορτώσει κάποια άλλη εφαρμογή. Η τακτική αυτή είναι συνηθισμένη σε χαμηλών δυνατοτήτων συστήματα, με λίγες μονάδες εισόδου – εξόδου και λίγους πόρους. Είναι μικρά, απλά και αξιόπιστα και είναι πολύ κατάλληλα για στατικές εφαρμογές όπως ο έλεγχος ενός φούρνου μικροκυμάτων ή ενός πλυντηρίου πιάτων.

Σε πολύπλοκες συσκευές, με πολλές μονάδες εισόδου εξόδου και απαιτητικά πρωτόκολλα επικοινωνίας, ο ρόλος του λειτουργικού είναι δύσκολος. Η κατασκευή ενός λειτουργικού, για κάθε συσκευή, γίνεται ασύμφορη. Επιπλέον, κάποια ενσωματωμένα συστήματα πρέπει να τρέχουν πολλές εφαρμογές, και να δίνουν στον χρήστη, τη δυνατότητα να τις διαχειρίζεται ή ακόμα και να εγκαθιστά ή να απεγκαθιστά κάποιες από αυτές. Αυτές είναι απαιτήσεις πολύπλοκων λειτουργικών συστημάτων που θυμίζουν έντονα λειτουργικά γενικού σκοπού.

Προκειμένου, να καλύψουν αυτές τις ανάγκες, κάποιες εταιρείες, δημιουργούν τα δικά τους ενσωματωμένα λειτουργικά, για να τα ενσωματώσουν στις συσκευές τους. Μια άλλη προσέγγιση είναι, η δημιουργία ενσωματωμένων συστημάτων γενικού σκοπού. Πρόκειται συνήθως για παραλλαγές γνωστών λειτουργικών συστημάτων όπως Linux, Windows, BSD, προσαρμοσμένες στις απαιτήσεις ενσωματωμένων συστημάτων.

2.6 Linux και ενσωματωμένα συστήματα

Το Linux είναι ένα λειτουργικό σύστημα τύπου UNIX βασισμένο στο μινιμαλιστικό λειτουργικό Minix του Andrew S. Tanenbaum. Δημιουργήθηκε το 1991 από τον Linus Torvalds, ο οποίος, τότε ήταν φοιτητής στο Πανεπιστήμιο του Ελσίνκι. Το δημοσίευσε με την άδεια ελεύθερου λογισμικού (2.7 Ελεύθερο Λογισμικό) GNU GPL. Καθώς το Linux είναι, ο πυρήνας του λειτουργικού συστήματος, συνδυάστηκε με το GNU project (περιγράφεται στο ελεύθερο λογισμικό) ώστε να αποτελέσει ολοκληρωμένο λειτουργικό.

Πλέον ο πυρήνας Linux αναπτύσσεται από πάρα πολλούς προγραμματιστές, είτε ερασιτέχνες, είτε στελέχη επιχειρήσεων που το στηρίζουν. Υπάρχουν πάρα πολλές διανομές του. Κάποιες από αυτές συντηρούνται από εταιρείες και άλλες από κάποια κοινότητα χρηστών. Υποστηρίζει πλήθος αρχιτεκτονικών επεξεργαστών και λόγω του ανοιχτού χαρακτήρα του, είναι εύκολο σε προγραμματιστές να το παραμετροποιήσουν, ώστε να καλύπτει τις ανάγκες οποιουδήποτε συστήματος. Έτσι το Linux τρέχει σε κάθε είδους υπολογιστικό σύστημα, προσωπικούς υπολογιστές, υπερυπολογιστές, ενσωματωμένα συστήματα, ακόμα και σε διαστημικό εξοπλισμό.

Όσον αφορά τα ενσωματωμένα συστήματα με Linux, μπορούμε να τα χωρίσουμε σε δύο κατηγορίες:

3. Τα ειδικού τύπου. Όπου ο πυρήνας προσαρμόζεται από τον παραγωγό του συστήματος ο οποίος προσαρμόσει ή αναπτύσσει και τις εφαρμογές. Ο χρήστης δεν έχει τη δυνατότητα να διαμορφώσει το λειτουργικό. Το μοντέλο αυτό είναι κατάλληλο για συστήματα με πολύ ξεκάθαρα καθορισμένο εύρος εργασιών, όπως φούρνοι μικροκυμάτων, ψηφιακές τηλεοράσεις, πλυντήρια.
4. Τα γενικού τύπου. Όπου ο πυρήνας και οι γύρω από αυτόν εφαρμογές διαμορφώνονται σε μορφή

διανομής. Αυτές οι διανομές αυτές είναι προσαρμοσμένες γενικά στις ανάγκες των λειτουργικών συστημάτων και δεν περιορίζονται σε ένα μόνο σύστημα. Ο χρήστης, έχει συνήθως τη δυνατότητα εγκατάστασης και απεγκατάστασης εφαρμογών και ποικίλες δυνατότητες παραμετροποίησης του λειτουργικού.

Περισσότερο ενδιαφέρον παρουσιάζει η δεύτερη κατηγορία, με πολλές εταιρείες να προσπαθούν να συντηρήσουν διανομές του είδους. Παραδείγματα τέτοιων διανομών είναι: MeeGo, webOS, Ångström distribution και γνωστότερη όλων το Android (Αν και το Android είναι βασισμένο στο Linux, αλλά υπάρχουν διαφωνίες, για το αν μπορεί να θεωρηθεί Linux). Αυτού του είδους οι διανομές είναι κατάλληλες για έξυπνα τηλέφωνα, υπολογιστές ταμπλέτες και άλλες συσκευές που πρέπει να προσφέρουν μεγάλο εύρος δραστηριοτήτων.

Η χρήση του Linux σε ενσωματωμένα συστήματα είναι φθηνή και τα παραγόμενα συστήματα αξιόπιστα. Ο πυρήνας και οι υπόλοιπες εφαρμογές του λειτουργικού μπορούν να προσαρμοστούν στις ανάγκες οποιασδήποτε συσκευής. Γιαυτό και η χρήση τους έχει εξαπλωθεί πολύ, και πολλές εταιρείες προτιμούν, από το να κατασκευάσουν από την αρχή ένα λειτουργικό σύστημα, να χρησιμοποιήσουν σα βάση το Linux, αναπτύσσοντας τη δική τους διανομή.

2.7 Ελεύθερο Λογισμικό

Η αρχή έγινε το 1983, όταν ο Richard Stallman ανακοίνωσε το GNU Project. Αρχικά, το λογισμικό του κάθε υπολογιστή, ήταν δύσκολο έως αδύνατο να τρέξει σε άλλα συστήματα, λόγω έλλειψης συμβατότητας. Όταν ο χώρος της πληροφορικής άρχισε να ωριμάζει, και εμφανίστηκαν υψηλού επιπέδου, για την εποχή, γλώσσες προγραμματισμού όπως η C, η διαλειτουργικότητα του λογισμικού αυξήθηκε σημαντικά. Τότε οι εταιρείες λογισμικού δημιούργησαν θεσμούς όπως το copyright, ώστε να αποτρέψουν τη διάδοση του λογισμικού τους χωρίς την άδεια τους.

Ο Stallman ήταν από τους πρώτους που κατάλαβαν την πορεία που είχε πάρει η αγορά λογισμικού και η πορεία αυτή, του φαινόταν ανυπόφορη. Όπως έχει πει (σε ελεύθερη μετάφραση από ομιλία του), "όταν έχεις προϊόν 'απαγορευμένου' (όρος που χρησιμοποιεί για το μη ελεύθερο λογισμικό) λογισμικού και το χρειάζεται ένας φίλος σου, έχεις δύο επιλογές:

5. Να του το δώσεις αθετώντας την άδεια χρήσης του
6. Να μη βοηθήσεις το φίλο σου. Και τα δύο είναι κακά επομένως είσαι σε αδιέξοδο.".

Αυτό και άλλα προβλήματα προσπαθεί να λύσει με το GNU Project και τον ορισμό του ελεύθερου λογισμικού.

Το ελεύθερο λογισμικό με βάση το Stallman πρέπει να σέβεται κάποιες βασικές ελευθερίες του χρήστη:

Ελευθερία 0: Η ελευθερία να χρησιμοποιήσει το πρόγραμμα όπως επιθυμεί.

Ελευθερία 1: Η ελευθερία να μελετήσει τον κώδικα του προγράμματος και να τον αλλάξει, αν επιθυμεί.

Ελευθερία 2: Η ελευθερία να αναδιανείμει αντίγραφα του προγράμματος.

Ελευθερία 3: Η ελευθερία να αναδημοσιεύσει αλλαγές και βελτιώσεις του προγράμματος, ώστε να ωφεληθεί η κοινότητα.

Καθώς οτιδήποτε πρέπει να έχει μια νομική υπόσταση στην κοινωνία που ζούμε, δημιουργήθηκε η άδεια GNU GPL(General Public License) δηλαδή η άδεια γενικής χρήσης του GNU. Εκδίδοντας ένα πρόγραμμα υπό αυτή την άδεια, ο δημιουργός του το θέτει υπό την υπηρεσία τις κοινότητας και το προστατεύει από επίδοξους που θα μπορούσαν να το οικειοποιηθούν σε βάρος των χριστών. Το GNU project είναι η συγκέντρωση προγραμμάτων από διάφορους προγραμματιστές, κυρίως ερασιτέχνες, με σκοπό τη δημιουργία ενός ελεύθερου λειτουργικού συστήματος.

Το 1991 αυτή η ουτοπία έγινε πραγματικότητα. Η κοινότητα του ελεύθερου λογισμικού είχε ήδη έτοιμα τα περισσότερα προγράμματα που αποτελούν ένα λειτουργικό, με εξαίρεση τον πυρήνα. Ο πυρήνας Hurd τον οποίο προσπαθούσαν να τελειοποιήσουν, απείχε πολύ από το να δουλέψει σωστά (και ακόμα απέχει). Τότε όμως ο Linus Torvalds εξέδωσε το Linux. Πείστηκε από την κοινότητα του GNU να τον εκδώσει υπό GNU GPL και από τότε υπάρχει ένα ολοκληρωμένο, ελεύθερο λειτουργικό σύστημα.

Από τότε πολλές εταιρείες και οργανισμοί ασχολούνται με το ελεύθερο λογισμικό, το οποίο έχει

συνεχώς ανοδική πορεία. Η άδεια GNU GPL έχει φτάσει στην έκδοση 3 (Το Linux kernel παρέμεινε στη δεύτερη έκδοση, λόγο διαφωνίας του Torvalds με κάποιες από τις αλλαγές) προκειμένου να αντεπεξέλθει στις νέες συνθήκες, ενώ άλλοι οργανισμοί έχουν τις δικές τους ελεύθερες άδειες όπως η Apache License. Οι άδειες αυτές αν και διατηρούν τη γενική ιδέα του ελεύθερου λογισμικού, διαφέρουν σε επιμέρους λεπτομέρειες.

Αξίζει να σημειωθεί ότι τα μέλη του ελεύθερου λογισμικού, έψαξαν τρόπους βιοπορισμού μέσα από το λογισμικό αυτό και υπάρχουν εταιρείες που επιβιώνουν από αυτό, κυρίως παρέχοντας υποστήριξη. Υπάρχουν και κάποιες διαφωνίες που οδήγησαν στη δημιουργία διαφορετικών αδειών χρήσης ελεύθερου λογισμικού. Μια πιο ακραία διαφωνία αφορά την ελεύθερη διανομή των προγραμμάτων. Κάποια μέλη της κοινότητας κατέληξαν ότι το σημαντικό είναι η πρόσβαση στον κώδικα του προγράμματος και οι υπόλοιπες ή κάποιες από τις υπόλοιπες ελευθερίες είναι δευτερεύουσες. Τα προγράμματα αυτής της φιλοσοφίας ονομάζονται λογισμικό ανοιχτού κώδικα.

Πλέων υπάρχουν πάρα πολλά προγράμματα ελεύθερου λογισμικού. Υπάρχουν άνθρωποι, αν και λίγοι, που χρησιμοποιούν εξολοκλήρου ελεύθερο λογισμικό. Όσο τρελό και αν ακουγόταν στην αρχή τελικά είναι εφικτό. Ακόμα και αν κάποιος δεν ενδιαφέρεται για αυτή την ιδεολογία, είναι σίγουρο ότι επωφελήθηκε από αυτό, καθώς μεγάλες εταιρείες κλειστού κώδικα, κάναν υποχωρήσεις και σε τιμές και σε ελευθερίες προς το χρήστη, εξαιτίας της πίεσης που δέχτηκαν από ανταγωνιστικά προϊόντα ελεύθερου λογισμικού.

Κεφάλαιο 3 – Διαδικασία Εκπόνησης

Παραλαμβάνοντας την πλακέτα, έλαβα δυστυχώς μόνο κινέζικο εγχειρίδιο. Η έντυπη βιβλιογραφία στο χώρο είναι περιορισμένη. Έτσι οι πηγές που χρησιμοποιήθηκαν είναι κυρίως σελίδες του διαδικτύου. Όσα εγχειρίδια ή βιβλία χρησιμοποιήθηκαν είναι σε ηλεκτρονική μορφή.

3.1 State of the Art

3.1.1 Για την επίτευξη των στόχων χρησιμοποιήθηκαν αρκετά προγράμματα και πρωτόκολλα. Κάποιες φορές υπήρχε δίλημμα ως προς το πιο είναι το καταλληλότερο. Αρχικά έγινε αντιληπτό ότι η προεπιλεγμένη διανομή ήταν ελλιπής και ο προεγκατεστημένος πυρήνας δεν περιείχε κάποιους χρήσιμους οδηγούς. Καθώς το supervivi που είναι ο προεπιλεγμένος boot loader δε δούλεψε με πυρήνα μεγαλύτερο των δυο MB, που ήταν απαραίτητος για την υποστήριξη αρκετών οδηγών. Επιλέγει ο u-boot σα βοηθητικός boot loader. Ο u-boot δίνει περισσότερες δυνατότητες, και τρέχει σα βοηθητικός του supervivi. Συγκεκριμένα ο supervivi είναι ο βασικός boot loader που εκτελείτε από τη μνήμη NOR. Έπειτα καλεί το u-boot από τη NAND σα να καλεί το λειτουργικό σύστημα. Το u-boot με τη σειρά του, τρέχει το λειτουργικό, είτε από βοηθητική μνήμη SD είτε μέσο δικτύου με τη χρήση NFS.

3.1.2 Μαζί με τη δυνατότητα να εγκαταστήσω οποιαδήποτε διανομή για ενσωματωμένα συστήματα ήρθε και το δίλημμα ποια είναι η πιο κατάλληλη. Οι τρεις δημοφιλέστερες διανομές για ενσωματωμένα συστήματα είναι οι μClinux (συχνά γράφετε ως uClinux), το emdebian (debian για ενσωματωμένα συστήματα) και το openembedded, buildroot.

Το emdebian είναι η έκδοση της δημοφιλέστατης διανομής debian για ενσωματωμένα συστήματα. Έχει όλα τα βασικά χαρακτηριστικά του debian αλλά είναι σημαντικά μικρότερο και ελαφρύτερο και υποστηρίζει πλήθος αρχιτεκτονικών. Σημαντική διαφορά είναι τα εργαλεία χτισίματος που δίνουν καλύτερο και λεπτομερέστερο έλεγχο, όσον αφορά τα πακέτα που θα εγκατασταθούν και τις απαιτήσεις τους. Έτσι δίνει ευελιξία στον σχεδιαστή του συστήματος για να το προσαρμόσει σε πολλές συσκευές και για διαφορετικές εργασίες.

Το buildroot είναι μια σειρά από makefiles και patches που κάνουν εύκολη τη δημιουργία ενός ενσωματωμένου συστήματος. Δίνει τη δυνατότητα να χτίσουμε όλα τα κομμάτια του συστήματος (πυρήνα, σύστημα αρχείων, boot-loader) είτε κάποιο από αυτά. Υποστηρίζει πλήθος πακέτων και αρχιτεκτονικών και υπόσχεται μια γρήγορη και σχετικά απλή διαδικασία.

Το μClinux έχει διαφέρει από τα υπόλοιπα, γιατί από μία ακόμα διανομή για ενσωματωμένα συστήματα, ενσωματώνει ένα παρακλάδι του linux πυρήνα, τροποποιημένο κατάλληλα για συσκευές χαμηλών προδιαγραφών.

Τέλος το openembedded είναι χαρακτηρίζετε περισσότερο ως framework για τη δημιουργία διανομής, παρά σα διανομή. Στοχεύει στα ενσωματωμένα συστήματα όπως μαρτυρά το όνομά του, δεν περιορίζετε όμως σε αυτά. Η όλη ιδέα είναι η χρήση του bitbake για το χτίσιμο συνταγών, η κάθε συνταγή χτίζει ένα πακέτο. Το bitbake ξεκίνησε από το gentoo, και η όλη διαδικασία χτισίματος μιας διανομής απο κώδικα παραπέμπει εκεί. Έχει και διαχειριστή πακέτων για έτοιμα μεταγλωττισμένα πακέτα (αρκεί να υπάρχουν τα κατάλληλα αποθετήρια). Τα πακέτα του σα μορφή μοιάζουν πολύ με αυτά του debian όπως και η λειτουργία του διαχειριστή τους.

Από τις παραπάνω επελέγη το openembedded γιατί μέσο της διανομής Angstrom υποστηρίζει τη mini2440, και έχει (συγκριτικά) καλή τεκμηρίωση και υποστήριξη για τη συγκεκριμένη πλακέτα. Έχει αποθετήρια που ταιριάζουν στην αρχιτεκτονική της, και online-builder που δίνει τη δυνατότητα να αποφύγουμε τη χρονοβόρα διαδικασία της μεταγλώττισης αν δε θέλουμε κάτι πολύ ειδικό.

3.1.3 Έχοντας έτοιμο για χρήση το λειτουργικό σύστημα, με του κατάλληλους οδηγούς και όλα τα απαραίτητα. Το επόμενο βήμα ήταν η επικοινωνία με τον αισθητήρα. Τα επικρατέστερα πρωτόκολλα ήταν το SPI, το I2C και η αναλογική επικοινωνία. Απέφυγα τον αναλογικό τρόπο, επειδή θα απαιτούσε χρήση του ADC (μετατροπέα από αναλογικό σε ψηφιακό). Ο εν λόγο μετατροπέας χρησιμοποιείται και από το καταδεικτικό αφής της οθόνης και η χρήση του θα μπορούσε να

δημιουργήσει προβλήματα, χωρίς να δίνει κάποιο σημαντικό πλεονέκτημα.

Το I2C είναι ένα πρωτόκολλο δημιουργημένο από τη Phillips, που συχνά αναφέρετε ως διεπαφή δύο καλωδίων. Το ένα καλώδιο είναι παλμός ρολογιού και το άλλο δεδομένα. Όπως γίνετε κατανοητό, τα δεδομένα στέλνονται αμφίδρομα. Το SPI από την άλλη, σχεδιάστηκε από τη Motorola, χρειάζεται τέσσερα καλώδια (υπάρχει και έκδοση με τρία). Το ένα καλώδιο είναι παλμός ρολογιού, το ένα αποστολή και το άλλο λήψη. Είναι επομένως αμφίδρομο πρωτόκολλο, και ταχύτερο από το I2C. Και τα δύο ακολουθούν το μοντέλο αφέντη - σκλάβου και υποστηρίζουν τη δυνατότητα ύπαρξης πολλών σκλάβων. Στο I2C η επιλογή σκλάβου γίνετε με βάση τη μοναδική του διεύθυνση, ενώ στο SPI γίνετε από το τέταρτο καλώδιο (χρειάζεται ένα ξεχωριστό καλώδιο για κάθε σκλάβο).

Αν και οποιοδήποτε από τα δύο πρωτόκολλα είναι κατάλληλο για το συγκεκριμένο σκοπό, επιλέχθηκε το SPI. Είναι αρκετά απλό σε συνδεσμολογία και προγραμματισμό, και καλύπτει με το παραπάνω τις απαιτήσεις της εργασίας.

3.1.4 Όταν η επικοινωνία με τον αισθητήρα θεωρήθηκε δεδομένη, και το πρόγραμμα λάμβανε κανονικά τις μετρήσεις, τέθηκε το θέμα της αποθήκευσης τους. Θα μπορούσαν να αποθηκευτούν είτε σε κάποιο αρχείο κειμένου, είτε σε βάση δεδομένων. Οι βάσεις δεδομένων που εξετάστηκαν είναι οι MySQL και SQLite. Είναι από τις πιο δημοφιλείς ελεύθερες προτάσεις. Δε χρειάζεται να δούμε πιο εξειδικευμένες λύσεις για μια απλή περίπτωση.

Το αρχείο κειμένου, αν και αποτελεί εφικτή λύση, έχει δυσκολίες στην υλοποίηση και η αποτελεσματικότητα του κρίνετε αμφίβολη. Η πολυπλοκότητας του αρχείου ως προς τη δομή δεν είναι μεγάλη. Αλλά θα έπρεπε να γραφούν μέθοδοι αναζητήσεις που με τη χρήση βάσης δεδομένων παρέχονται έτοιμες. Το σημαντικότερο όμως είναι ότι οι βάσεις δεδομένων είναι δοκιμασμένες σε πολύ πιο απαιτητικές χρήσεις. Έτσι μπορούμε να είμαστε σίγουροι ότι θα καλύψουν της ανάγκες μας. Αντίθετα η σχεδίαση του αρχείου κειμένου θα έπρεπε να γίνει προσεκτικά για να μην προκύψουν αργότερα απρόβλεπτες καταστάσεις, όπως έλλειψη δυνατοτήτων παραμετροποίησης.

Η MySQL είναι η πιο διαδεδομένη βάση δεδομένων. Η αποτελεσματικότητα της είναι αναμφίβολη, και συνεπώς αφήνει λίγα περιθώρια να ψάξουμε κάτι άλλο. Παρόλα αυτά η SQLite3 κρίθηκε πιο κατάλληλη για τη συγκεκριμένη εφαρμογή. Πρώτον γιατί δεν ακολουθεί το πρωτόκολλο πελάτη εξυπηρετητή, αλλά είναι ενσωματωμένη στο πρόγραμμα μας. Αυτό μειώνει της απαιτήσεις του συστήματος, αφού δε χρειάζεται να τρέχει μια διεργασία εξυπηρετητή για τη βάση δεδομένων. Δεύτερον, αν και έχει λίγο λιγότερες δυνατότητες από την MySQL, είναι υπέρ επαρκής για μια βάση δεδομένων με ένα μόνο πίνακα. Μάλιστα ίσως ήταν υπερβολή η χρήση της MySQL για κάτι τόσο απλό. Τέλος είναι εύκολη στη χρήση και διαθέσιμη μέσω πολλών γλωσσών προγραμματισμού. Προσωπικά εκτίμησα το ότι κάθε βάση δεδομένων είναι ένα και μόνο αρχείο, που δημιουργείτε και φυλάσσετε όπου θέλουμε. Αυτός ο κομψός τρόπος αποθήκευσης και πρόσβασης στα δεδομένα είναι πολύ βολικός.

3.1.5 Η εφαρμογή λήψης και αποθήκευσης των μετρήσεων γράφηκε στη C. Είναι μια γλώσσα γνωστή, με τουλάχιστο επαρκείς δυνατότητες και εύκολη πρόσβαση στον μεταγλωττιστή για ARM. Επιπλέον έχει βιβλιοθήκες και καλή τεκμηρίωση για SQLite. Δεν υπήρξε ανάγκη αναζήτησης άλλης γλώσσας.

3.1.6 Η ανάγνωση των δεδομένων θα μπορούσε να υλοποιηθεί είτε με μία διεπαφή γραμμής εντολών, είτε με διαδικτυακή διεπαφή. Η διαδικτυακή διεπαφή κρίθηκε προτιμότερη, καθώς έχει αρκετά πλεονεκτήματα. Δίνει εξ ορισμού πρόσβαση σε απομακρυσμένα συστήματα, χωρίς να απαιτεί εξειδικευμένο λογισμικό. Και είναι πολύ πιο εύχρηστη και ευχάριστη στη χρίση από τη γραμμή εντολών. Οπότε απαραίτητος είναι και ένας web server.

Οι επικρατέστεροι για χαμηλών επιδόσεων συστήματα, είναι οι boa(ήταν προεγκατεστημένος), cherokee, lighttp, foxserve. Ο boa αν και προεγκατεστημένος στην προεπιλεγμένη διανομή, δε δίνει πολλές δυνατότητες. Στα περισσότερα forum συνιστάτε η αντικατάστασή του αν θέλουμε δυνατότητες. Παρόλα αυτά είναι σταθερός, και τις όποιες δυνατότητες έχει τις αξιοποιεί σωστά. Ο lighttp είναι ο πιο εύκολα υποστηριζόμενος από το openembedded αλλά δεν προσφέρει κάτι παραπάνω από τον boa. Ο foxserv είναι αρκετά υποσχόμενος. Είναι μια τροποποιημένη έκδοση του Apache. Στη σελίδα του λέει ότι υποστηρίζει php, SQLite και πολλά άλλα. Ωστόσο δεν είδα αναφορές για αυτόν σε σελίδες σχετικές με την πλακέτα, και δεν υποστηρίζεται από το

openembedded, οπότε η εγκατάστασή του ίσως αποδεικνύονταν δύσκολη.

Ο Cherokee που αποτελεί και την τελική επιλογή, είναι ένας ελαφρύς web server. Έχει πολλές δυνατότητες και υποστηρίζετε από το openembedded. Αν και η έκδοση αποθετήρια του openembedded που είναι συμβατή με την αρχιτεκτονική της πλακέτας είναι παλιά, δουλεύει και καλύπτει της ανάγκες της εργασίας.

Εδώ πρέπει να αναφερθεί το δίλημμα php ή cgi. Η αρχική σκέψη ήταν η διαδικτυακή διεπαφή να γραφεί σε php. Βλέποντας όμως τα προβλήματα πολλών web server να την υποστηρίξουν και αναζητώντας άλλους τρόπους, ήρθα σε επαφή με το cgi. Η php αναμφίβολα έχει πολλά πλεονεκτήματα, γιαυτό και επικράτησε στον προγραμματισμό διαδικτύου. Το cgi όμως δίνει τη δυνατότητα να γράψουμε τη διαδικτυακή μας εφαρμογή σε όποια γλώσσα θέλουμε. Γράφοντας λοιπών τα cgi αρχεία στη C, αφενός κερδίζουμε σε επιδώσεις (αφού η εφαρμογή μας είναι εκτελέσιμο αρχείο), και αφετέρου χρησιμοποιούμε μία μόνο γλώσσα προγραμματισμού για όλη την εργασία. Επιπλέον η χρήση των cgi θα ήταν πιθανό αναπόφευκτη για να υλοποιηθούν οι ενέργειες που μπορεί να εκτελέσει ο χρήστης μέσα από τη διαδικτυακή εφαρμογή.

Μια επιπλέον πιθανή λύση, ήταν το python bottle. Ένα ελαφρύ framework της python κατάλληλο για εφαρμογές διαδικτύου, που δε χρειάζεται ξεχωριστό web server. Ήταν μια ενδιαφέρουσα πρόταση, ωστόσο έχει κάποια μειονεκτήματα. Το εν λόγο πακέτο είναι ακόμα σε πρώιμη έκδοση και συνεπώς δεν είμαι σίγουρος για τη σταθερότητα του. Επιπλέον δεν ξέρω τις δυνατότητες του και είναι αμφίβολο αν το τελικό αποτέλεσμα θα ήταν το ίδιο ικανοποιητικό με τα C cgi. Τέλος το ότι δεν βρίσκετε στα αποθετήρια από το openembedded είναι άλλος ένας αποθαρρυντικός παράγοντας.

3.2 Σημαντικοί στόχοι για την ολοκλήρωση της πτυχιακής

7. Εγκατάσταση boot-loader
8. Εγκατάσταση λειτουργικού
9. Επιλογή πρωτοκόλλου επικοινωνίας
10. Ανάλυση δυνατοτήτων SPI
11. Προγράμματος λήψης μετρήσεων
12. Ενεργοποίηση web server
13. Συγγραφή διεπαφής
14. Διόρθωση προγράμματος λήψης μετρήσεων και διεπαφής, ώστε να συνεργάζονται σωστά
15. Συγγραφή αναφοράς

3.3 Υλικό που χρησιμοποιήθηκε

3.3.1 Πλακέτα FriendlyARM mini2440

Η mini2440 είναι ένας υπολογιστής πλακέτα της οικογένειας FriendlyARM. Χρησιμοποιεί τον επεξεργαστή S3C2440 της Samsung που κυκλοφόρησε το 2003. Ένας τριανταδιάμπιτος RISC επεξεργαστής. Ο πυρήνας του οποίου είναι ο ARM920T και χρησιμοποιεί το σετ εντολών ARMv4T. Αναλυτικά τα χαρακτηριστικά της πλακέτας είναι είναι:

Διαστάσεις: 100x100

Επεξεργαστής: 400 MHz Samsung S3C2440A ARM920T (μέγιστη συχνότητα 533 MHz)

Μνήμη RAM: 64 MB SDRAM, 32 bit Bus

Μνήμη flash: 128 MB NAND Flash και 2 MB NOR Flash with BIOS (υπάρχουν εκδώσεις με 64 MB, 256 MB, 1GB NAND Flash επίσης)

Εξωτερική μνήμη: SD-Card

Serial Ports: 1x DB9 connector (RS232), total: 3x serial port connectors

USB: 1x USB-A Host 1.1, 1x USB-B Device 1.1

Έξοδος ήχου: 3.5 mm stereo jack

Είσοδος ήχου: Connector + Condenser microphone

Ethernet: RJ-45 10/100M (DM9000)

RTC: Real Time Clock with battery (CR1220)

Beeper: PWM buzzer

Camera: 20 pin (2.0 mm) Camera interface (Η κάμερα έρχεται χωριστά)

LCD Interface: 41 pin (1.0 mm) connector for FriendlyARM Displays and VGA Board (με οθόνη 3.5" LCD)

Touch Panel: 4 wire resistive

User Inputs: 6x push buttons and 1x A/D pot

User Outputs: 4x LEDs

Expansion: 40 pin System Bus, 34 pin GPIO, 10 pin Buttons (2.0 mm)

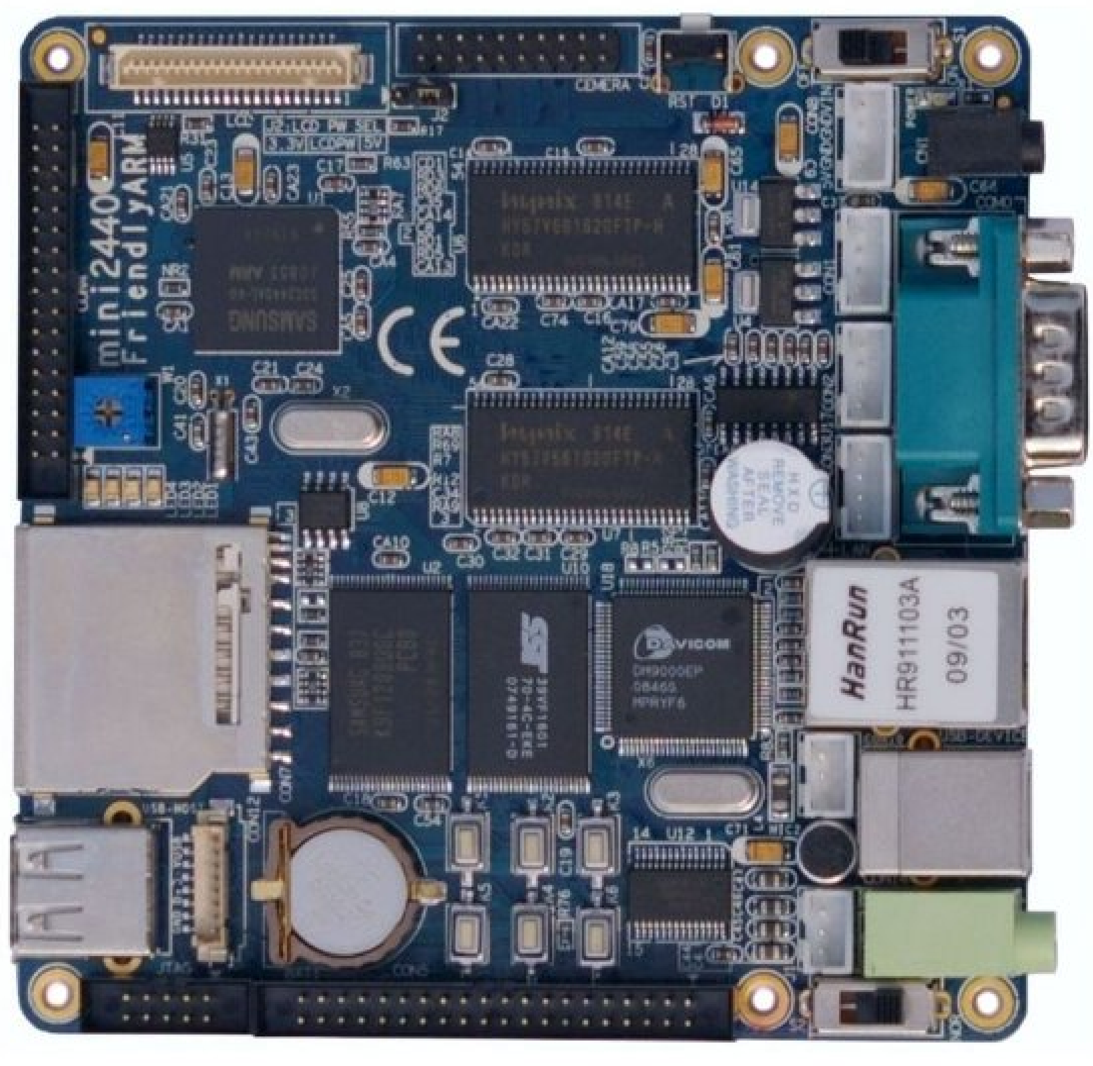
Debug: 10 pin JTAG (2.0 mm)

Power: regulated 5V (DC-Plug: 1.35mm inner x 3.5mm outer diameter)

Κατανάλωση ισχύος: Mini2440: 0.3 A, Mini2440 + 3.5" LCD: 0.6 A, Mini2440 + 7" LCD: 1 A

Υποστηριζόμενα λειτουργικά συστήματα: Linux 2.6, Windows CE, Android

Η πλακέτα είναι εξοπλισμένη με δύο μη πτητικές μνήμες. Η μνήμη NAND, την οποία συχνά αναφέρουμε σα μνήμη flash και χρησιμοποιείτε ως κύριο αποθηκευτικό μέσο του συστήματος. Η μνήμη NOR είναι πολύ μικρότερη και εκεί είναι εγκατεστημένος ο boot-loader. Σε μικρές συσκευές όπως αυτή, συνήθως απουσιάζει κάποιο σύστημα διαχείρισης εισόδου εξόδου όπως το bios. Ο boot-loader επομένως είναι απαραίτητος για την εκκίνηση του λειτουργικού συστήματος.



Εικόνα 2: mini2440

3.3.2 Αισθητήρας

Ο αισθητήριας θερμότητας και υγρασίας που χρησιμοποιήθηκε είναι ο DC-SS500 της Sure Electronics . Ενσωματώνει το κύκλωμα επεξεργασίας PIC16F690 και τον αισθητήρα HS1101 . Έτσι μπορεί να λειτουργήσει απρόσκοπτα σε μεγάλο εύρος περιβαλλοντικών συνθηκών και παρέχει αρκετές διεπαφές επικοινωνίας για να επιλέξουμε την κατάλληλη.



Συγκεκριμένα:

Μπορεί να λειτουργήσει είτε με 3,3V είτε με 5V. Μπορεί να λειτουργήσει μεταξύ -15 ºC με 60 ºC και για υγρασία 1 – 99 % RH. Στις μετρήσεις ενδέχεται να υπάρχει μία απόκλιση. Η απόκλιση θερμοκρασίας είναι τυπικά 2 βαθμοί και σε ακραίες περιπτώσεις έως και 5. Αντίστοιχά απόκλιση υγρασίας είναι 5% με 10% σε σπάνιες περιπτώσεις, ενώ πιο ακριβή αποτελέσματα έχομε για τιμές 10 – 95% υγρασία σε θερμοκρασίες 0 – 60 ºC. Ο χρόνος απόκρισης είναι 10 δευτερόλεπτα για τη μέτρηση της θερμοκρασίας και 6 για τη μέτρηση υγρασίας. Οι τρόποι επικοινωνίας που υποστηρίζονται είναι: αναλογικά γραμμικό σήμα εξόδου για θερμοκρασία και υγρασία, είτε ψηφιακά με UART half duplex ή SPI slave mode . Πιο βολικός για τη συγκεκριμένη εργασία κρίθηκε ο τελευταίος τρόπος, με SPI.

Κεφάλαιο - 4 Λειτουργικό και λογισμικό

Κατά το ξεκίνημα της εργασίας, έγινε γρήγορα αντιληπτό, ότι η πρόσβαση σε ακριβείς πληροφορίες σχετικές με το θέμα θα ήταν προβληματική. Άτομα που ασχολούνται συστηματικά, είναι συνήθως επαγγελματίες που δε συχνάζουν σε "χώρους" ανταλλαγής πληροφοριών. Λίγοι ασχολούνται συστηματικά με ενσωματωμένα συστήματα για χόμπι ή έρευνα, και αυτά τα συστήματα διαφέρουν μεταξύ τους, οπότε ήταν σχεδόν αδύνατη η επαφή με πεπειραμένα άτομα που θα μπορούσαν να προσφέρουν καθοδήγηση. Έπρεπε λοιπόν να ξεκαθαρίσει το τοπίο και να αποκτήσω μια σαφή άποψη του τη χρειάζομαι και τι μπορώ να κάνω.

Το κύριο μέρος της δουλειάς ήταν οι πειραματισμοί με την πλακέτα. Μέσα από αυτούς επιλέχθηκε το κατάλληλο λογισμικό (εκκινητής (boot-loader), πυρήνας, λειτουργικό, κτλ). Έγιναν επίσης αντιληπτές οι δυνατότητες της πλακέτας και του συμβατού λογισμικού. Όταν επιτεύχθηκε αυτό, ήταν δυνατό να καθοριστούν οι απαιτήσεις συστήματος, και ορίστηκαν οι ακριβείς στόχοι της εργασίας. Σε αυτό το κεφάλαιο θα ασχοληθούμε με τις δοκιμές, πετυχημένες και αποτυχημένες, που χρειάστηκαν για την απόκτηση εμπειρίας με την πλακέτα mini2440. Κυρίως όμως με το λογισμικό που τελικά χρησιμοποιήθηκε.

Όσον αφορά τον κύριο υπολογιστή που χρησιμοποίησα. Χρησιμοποιήθηκαν ανά διαστήματα ο σταθερός ή ο φορητός μου υπολογιστής, πάντα σε περιβάλλων Linux. Τα πρώτα βήματα έγιναν σε Ubuntu 9.04, ενώ το λειτουργικό μέρος ολοκληρώθηκε σε Ubuntu 10.04 (κατόπιν αναβάθμισης). Πληροφορίες για της απαιτήσεις σε λογισμικό δίνονται στις επιμέρους ενότητες.

4.1 Αρχική κατάσταση

Παραλαμβάνοντας την πλακέτα, τη βρήκα εξοπλισμένη με μια διανομή Linux παραμετροποιημένη από τη FriendlyARM. Η διανομή περιελάμβανε γραφικό περιβάλλων Qtopia (ένα γραφικό περιβάλλων για υπολογιστές παλάμης) με κάποια βασικά προγράμματα εγκατεστημένα. Το Qtopia είναι ένα ελαφρύ γραφικό περιβάλλον, φτιαγμένο για μικρές συσκευές αφής. Στα προεγκατεστημένα προγράμματα συμπεριλαμβάνονταν πρόγραμμα προβολής εικόνων, αναπαραγωγής ήχου και βίντεο, επεξεργαστή κειμένου, εικονικό πληκτρολόγιο, κονσόλα, ακόμα και κάποια παιχνίδια. Υπήρχαν βέβαια προγράμματα ρυθμίσεων, καθώς και κάποιες εφαρμογές που έκαναν χρήση των LED. Εντύπωση μου έκανε ο προεγκατεστημένος boa εξυπηρετητής ιστού (web server). Στη σελίδα που έτρεχε, παρείχε τη δυνατότητα ελέγχου των LED με τη χρήση cgi.

Το προεγκατεστημένο πακέτο βέβαια είχε κάποια προβλήματα. Κατ' αρχήν δεν υπήρχαν οι κατάλληλοι οδηγοί (drivers) στον πυρήνα για την υποστήριξη GPIO και SPI. Επίσης όπως διαπίστωσα με λίγα πειράματα και ψάξιμο στο διαδίκτυο, η διανομή δίνει πολύ λίγες δυνατότητες παραμετροποίησης. Ο προεπιλεγμένος boot-loader (supervivi) δε βρισκόταν σε καλύτερη κατάσταση. Η πρόταση που βρήκα σε αρκετές σελίδες ήταν η αντικατάσταση όλων.

4.2 Πυρήνας

Η αρχική σκέψη ήταν πως, αρκούσε η αντικατάσταση του πυρήνα, με έναν καλύτερα εξοπλισμένο σε οδηγούς συσκευών. Κάτι τέτοιο όμως αποδείχθηκε αδύνατο. Έχτισα ένα πυρήνα με βάση τις οδηγίες του χρήστη emeb στο forum της FriendlyARM. Για να έχει όμως όλα τα απαραίτητα, το μέγεθός του ήταν μεγαλύτερο από το μέγιστο που μπορούσε να διαχειριστεί το supervivi. Αυτό έκανε απαραίτητη την αλλαγή του Boot-loader και τελικά και της διανομής.

Στα επόμενα υποκεφάλαια (4.2.*) αναλύονται οι τρόποι του Emeb και του Bill για το χτίσιμο πυρήνα. Καθώς και οι δύο τρόποι έχουν πλεονεκτήματα και μειονεκτήματα, παρουσιάζεται και ο υβριδικός τρόπος που τελικά χρησιμοποίησα.

4.2.1 Ο τρόπος του Emeb

Η πρώτη προσπάθεια (και η πρώτη μου προσπάθεια να στήσω linux kernel γενικότερα), έγινε ακολουθώντας της οδηγίες του χρήστη Emeb, από μια σελίδα που δεν είναι πλέον σε λειτουργία. Βέβαια καθώς όταν βρήκα αυτές τις πληροφορίες πολύτιμες έσωσα τη σελίδα, καθώς και τα αρχεία που χρειάζονταν να κατεβάσει κανείς για να ακολουθήσει τις οδηγίες. Έτσι ήμουν σίγουρος ότι δε θα

τις χάσω. Τη φιλοξενώ ως συμπιεσμένο αρχείο στη θέση: https://dl.dropbox.com/u/6983593/emeb_kernel.zip προκειμένου να εξυπηρετήσω και άλλους χρήστες που ψάχνανε τις ίδιες πληροφορίες.

Διαδικασία:

Κατεβάζουμε από τη σελίδα της πλακέτας (http://www.friendlyarm.net/downloads) τον cross-compiler. Σε ubuntu, τα οποία είναι η διανομή που δουλεύω, χρειαζόμαστε εγκατεστημένο το ncurses5. Έπειτα κατεβάζουμε τον κώδικα του πυρήνα είναι επίσης από την επίσημη σελίδα (http://www.friendlyarm.net/dl.php?file=linux-2.6.32.2-mini2440_20100113.tgz).

Αποσυμπιέζουμε τον πυρήνα (μπορούμε να το κάνουμε με την εντολή tar -zxvf linux-2.6.32.2-mini2440_20100113.tgz). Για να τον χτίσουμε με της προεπιλεγμένες ρυθμίσεις ακολουθούμε τα παρακάτω τέσσερα βήματα.

16. cd linux-2.6.32.2 (μπαίνουμε στο φάκελο όπου μόλις αποσυμπιέσαμε τον πυρήνα)
17. cp config_mini2440_t35 .config (αντιγράφουμε το έτοιμο αρχείο ρυθμίσεων που ταιριάζει στην οθόνη της πλακέτας μας. config_mini2440_t35 στην περίπτωση μου)
18. make menuconfig (εκτελούμε την εντολή για να δημιουργηθούν τα απαραίτητα αρχεία και βγαίνουμε)
19. make zImage (και περιμένουμε να τελειώσει η διαδικασία – παίρνει αρκετή ώρα)

πριν από αυτό πρέπει να έχουμε προσθέσει στο PATH τη θέση του cross-compiler (export PATH="${PATH}:'Θέση του cross-compiler'/arm-2008q3/bin")

Η εικόνα του πυρήνα που μόλις δημιουργήσαμε βρίσκεται στο φάκελο “./arch/arm/boot”.

Το κύριο πλεονέκτημα του να χτίσουμε το δικό μας πυρήνα είναι ότι μπορούμε να προσθέσουμε τα χαρακτηριστικά που επιθυμούμε. Οπότε συνεχίζουμε τη διαδικασία, αλλάζοντας τις προεπιλεγμένες ρυθμίσεις. Ο πυρήνας που κατεβάσαμε είναι παραμετροποιημένος για mini2440 και επομένως περιέχει οδηγούς για UARTs, LCD, Ethernet, NAND, MMC και I2C. Επίσης το αρχείο αρχικοποίησης του πυρήνα είναι σωστά ρυθμισμένο για να λειτουργήσουν.

Προκειμένου να προσθέσουμε υποστήριξη για SPI πρέπει να τροποποιήσουμε το αρχείο “linux-2.6.32.2/arch/arm/mach-s3c2440/mach-mini2440.c”. Ευτυχώς ο Emeb ανέβασε το δικό του τροποποιημένο αρχείο. Το μόνο που χρειάζεται είναι να το αντικαταστήσουμε με το υπάρχων. Επίσης πρέπει να προσθέσουμε τους οδηγούς τους στο “.cofig”. Και εδώ ο πιο απλός τρόπος είναι να πάρουμε το έτοιμο αρχείο του Emeb “cp config_mini2440_t35_gpio_spi .config”. Για να λειτουργήσει το εν λόγω αρχείο πρέπει η πλακέτα να έχει οθόνη T35 LCD.

Σε περίπτωση που έχουμε άλλη οθόνη πρέπει να περιηγηθούμε στο menuconfig και να ενεργοποιήσουμε τα παρακάτω:

GPIO sysfs
SPI
SPI Master
SPIDEV
SPI S3C24XX

Πρέπει να βρίσκονται κάτω από το μενού devices.

Επίσης μέσω menuconfig μπορούμε να προσθέσουμε υποστήριξη για περισσότερες συσκευές ή χαρακτηριστικά, όπως συστήματα αρχείων.

Πριν από κάθε χτίσιμο του πυρήνα με νέο αρχείο ρυθμίσεων, συνιστάται να εκτελούμε την εντολή “make clean”.

Ο τρόπος του Emeb ήταν αποτελεσματικός και έδινε λειτουργικούς πυρήνες. Τον χρησιμοποίησα για αρκετό καιρό με το openembedded (build 01.2008). Όταν πήγα να περάσω όμως την καινούρια έκδοση του openembedded (build 03.2011), κατά τη διαδικασία της εκκίνησης έπαιρνα το μήνυμα "/dev/misc/rtc: file not fount". Τη λύση έδωσε ένας συνδυασμός του τρόπου τού Emeb, με τον τρόπο του Bill.

4.2.2 Ο τρόπος του bill

Σε αντίθεση με τον Emeb που χρησιμοποιεί συγκεκριμένο αρχείο για πυρήνα, ο Bill προτείνει τη χρήση αποθετηρίου git. Έτσι παίρνουμε την πιο ενημερωμένη έκδοση του πυρήνα. Μπορούμε να

κατεβάσουμε την τελευταία έκδοση από το "http://kernel.org", έτσι όμως δε θα είχαμε όλους τους οδηγούς και τις παραμετροποιήσεις για να δουλέψουν όλες οι συσκευές της πλακέτας. Οπότε παίρνουμε τον κώδικα από το αποθετήριο "git://repo.or.cz/linux-2.6/mini2440.git". Το οποίο προορίζεται για τη mini2440.

Τα βήματα είναι τα εξής:

20. Δημιουργούμε ένα φάκελο για τον νέο πυρήνα "mkdir mini2440-kernel"
21. Κατεβάζουμε ένα αντίγραφο του αποθετηρίου git "git clone git://repo.or.cz/linux-2.6/mini2440.git"
22. Μπαίνουμε στο φάκελο που δημιουργήθηκε "cd mini2440"
23. Έπειτα για ένα πυρήνα με της προεπιλεγμένες ρυθμίσεις
24. Δημιουργούμε ένα αρχείο προεπιλεγμένων ρυθμίσεων "CROSS_COMPILE='θέση'/arm-2008q3/bin/arm-none-linux-gnueabi- ARCH=arm make mini2440_defconfig"
25. Χτίζουμε τον πυρήνα "CROSS_COMPILE='θέση'/arm-2008q3/bin/arm-none-linux-gnueabi- ARCH=arm make"
26. Έπειτα εγκαθιστούμε τα module στο σύστημα αρχείων που θα χρησιμοποιήσουμε "CROSS_COMPILE='θέση'/arm-2008q3/bin/arm-none-linux-gnueabi- ARCH=arm INSTALL_MOD_PATH=/mnt make modules_install "

Όπου θέση βέβαια γράφουμε τη θέση του cross-compiler μας. Η διαδικασία για να αποκτήσουμε ένα σύστημα αρχείων περιγράφεται αργότερα (4.5 Openembedded).

4.2.3 Τελική λύση

Προκειμένου να έχουμε τα πλεονεκτήματα και τον δύο μεθόδων, κατεβάζουμε τον πυρήνα μέσω git όπως ακριβώς προτείνει ο Bill, και προσθέτουμε τις ρυθμίσεις του Emeb για την υποστήριξη SPI. Για να το κάνουμε αυτό:

27. Ακολουθούμε τη μέθοδο του Bill μέχρι το βήμα 4.
28. Έπειτα εκτελούμε "menuconfig" και ενεργοποιούμε ότι προτείνει ο Emeb (Τα περισσότερα είναι στο μενού devices)
29. προαιρετικά προσθέτουμε και άλλα χαρακτηριστικά προσωπικά πρόσθεσα υποστήριξη για ext3, ext4, NFS και Kernel .conf για να μπορώ να ανακτήσω τις ρυθμίσεις του πυρήνα
30. Τέλος προσθέτουμε στο αρχείο "arch/arm/mach-s3c2440/mach-mini2440.c" τις απαραίτητες ρυθμίσεις για τους ακροδέκτες της SPI

Για να το κάνουμε αυτό:

- μπαίνουμε στο φάκελο "arch/arm/mach-s3c2440" (cd arch/arm/mach-s3c2440)
- προαιρετικά δημιουργούμε ένα αντίγραφο ασφαλείας για το mach-mini2440.c (cp mach-mini2440.c mach-mini2440.c_bak)
- Προσθέτουμε τις βιβλιοθήκες linux/spi/spi.h και mach/spi.h

```
#include <linux/spi/spi.h>
#include <mach/spi.h>
```

- Λίγο πιο κάτω, εγώ το έβαλα κάτω από το /* touchscreen configuration */ προσθέτουμε

```
/* SPI driver info */
static struct spi_board_info __initdata mini2440_spi_board_info[]
= {
    // SPI Port 0
    {
        .modalias       = "spidev",
        .max_speed_hz    = 48000000, //48 Mbps
        .bus_num       = 0,
        .chip_select    = 0,
        .mode           = SPI_MODE_1,
    },
};
```

```
  static void mini2440_spi0_cs(struct s3c2410_spi_info *spi, int cs,
int pol)
  {
      s3c2410_gpio_setpin(S3C2410_GPG(2), pol);
  }

  static struct s3c2410_spi_info mini2440_spi0_platdata = {
      .num_cs = 1,
      .bus_num = 0,
      .set_cs = mini2440_spi0_cs,
  };
```

- Μέσα στο “static struct platform_device *mini2440_devices[] = {” προσθέσουμε

```
&s3c_device_spi0,
```

- Τέλος μέσα στη συνάρτηση “static void __init mini2440_init(void)” προσθέσουμε

```
    /* setup SPI0 pins */
    s3c2410_gpio_cfgpin(S3C2410_GPE(11), S3C2410_GPE11_SPIMISO0);
    s3c2410_gpio_cfgpin(S3C2410_GPE(12), S3C2410_GPE12_SPIMOSI0);
    s3c2410_gpio_cfgpin(S3C2410_GPE(13), S3C2410_GPE13_SPICLK0);
    s3c2410_gpio_cfgpin(S3C2410_GPG(2), S3C2410_GPIO_OUTPUT);
    s3c_device_spi0.dev.platform_data = &mini2440_spi0_platdata;
    spi_register_board_info(mini2440_spi_board_info,
       ARRAY_SIZE(mini2440_spi_board_info));
    printk(KERN_ERR "Setup SPI0\n");
```

- Σώνουμε στο αρχείο και βγαίνουμε.
- Επιστρέφουμε στον αρχικό φάκελο του πυρήνα (τον mini2440)
- Τέλος χτίζουμε κανονικά τον πυρήνα και εγκαθιστούμε τα modules όπως στα βήματα 5 και 6 της μεθόδου του Bill.

Συνιστάτε make clean, κάθε φορά που αλλάζουμε ρυθμίσεις.

4.3 Supervivi

Μικρές συσκευές όπως η FriendlyARM, δεν περιλαμβάνουν BIOS. Έτσι ο boot-loader παίζει το ρόλο του BIOS και είναι απαραίτητο στοιχείο για την εκκίνηση του λειτουργικού. Ο προεπιλεγμένος boot-loader είναι ο supervivi. Εκτός από της βασικές λειτουργίες ενός boot-loader, χρησιμεύει για να διαχειριστούμε της κατατμήσεις της μνήμης NAND, τη μεταφορά του πυρήνα και του συστήματος αρχείων στις κατατμήσεις και την πραγματοποίηση ρυθμίσεων που αφορούν την εκκίνηση της συσκευής.

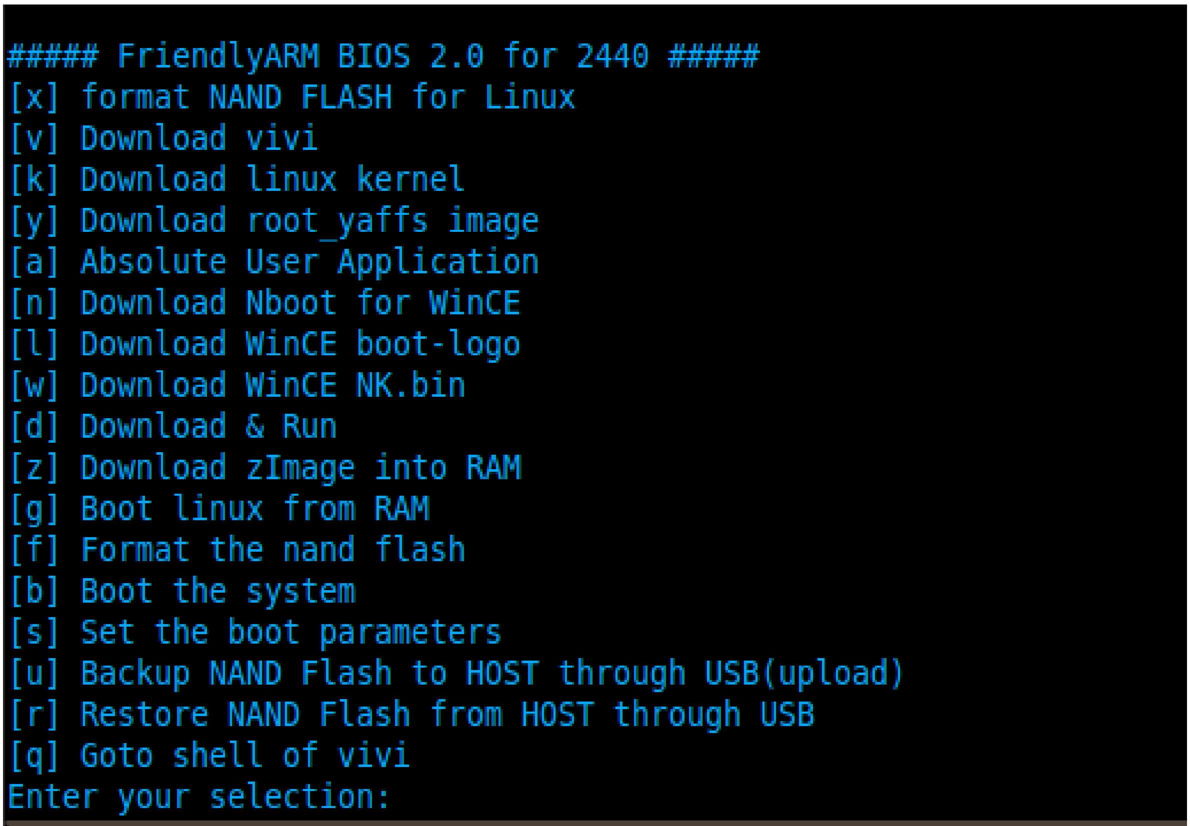
Καθώς το πρώτο σχέδιο όπως αναφέρθηκε στην ενότητα (4.1 Αρχική κατάσταση) ήταν να αντικατασταθεί μόνο ο πυρήνας, η αντικατάστασή του θα γινόταν με το supervivi. Η κατάτμηση όμως για τον πυρήνα ήταν 2.4MB ενώ ο καινούριος πυρήνας (με τη μέθοδο του Emeb) 2.6MB. Προχώρησα λοιπόν σε ανακατάτμηση της NAND. Πέρασα το νέο πυρήνα και το υπάρχον σύστημα αρχείων, κατεβασμένο εκ νέου από τη σελίδα της FriendlyARM.

Προκειμένου να χειριστούμε το supervivi, πρέπει να συνδέσουμε την πλακέτα με έναν υπολογιστή μέσω σειριακής θύρας. Για να μεταφέρουμε αρχεία, πρέπει να συνδέσουμε και ένα καλώδιο USB. Υπάρχει τρόπος ωστόσο να μεταφέρουμε αρχεία και μέσω tftp, αλλά κάτι τέτοιο απαιτεί εγκατάσταση tftp server στον κύριο υπολογιστή και είναι πιο πολύπλοκο.

Το προτεινόμενο εργαλείο χειρισμού του supervivi είναι το DNW, και οι οδηγίες είναι γραμμένες για αυτό το πρόγραμμα. Το DNW βέβαια είναι εκτελέσιμο για Windows. Δεδομένου ότι σκοπός της συγκεκριμένης εργασίας είναι να ασχοληθεί με λύσεις βασισμένες αμιγώς σε Linux, αναζήτησα τα ανάλογα προγράμματα για linux. Δεν υπάρχει βέβαια κάτι τόσο συμπαγές όσο το DNW που να ενσωματώνει όλες τις απαραίτητες λειτουργίες. Η ίδια δουλειά όμως μπορεί να γίνει με ένα πρόγραμμα εξομοίωσης τερματικού όπως το picocom ή το minicom και το πρόγραμμα usbpush από τη σελίδα της FriendlyARM. Χρησιμοποίησα το picocom που είναι πιο ελαφρύ, επειδή δε χρειαζόμαστε περισσότερες δυνατότητες από αυτές που δίνει.

Προκειμένου να κάνουμε οποιαδήποτε αλλαγή στις ρυθμίσεις του supervivi:

- Ανοίγουμε ένα τερματικό και εκτελούμε την εντολή (picocom -b 115200 /dev/ttyS0 --send-cmd "sx -vv"), υποθέτοντας ότι η πλακέτα είναι συνδεδεμένη στην πρώτη σειριακή θύρα του υπολογιστή.
- επιλέγουμε τη NOR μνήμη (3.3.1 Πλακέτα FriendlyARM mini2440 τελευταία παράγραφος) από το διακόπτη και ενεργοποιούμε την πλακέτα. Κατά τη μετακίνηση του διακόπτη η πλακέτα πρέπει να είναι απενεργοποιημένη.


```
##### FriendlyARM BIOS 2.0 for 2440 #####
[x] format NAND FLASH for Linux
[v] Download vivi
[k] Download linux kernel
[y] Download root_yaffs image
[a] Absolute User Application
[n] Download Nboot for WinCE
[l] Download WinCE boot-logo
[w] Download WinCE NK.bin
[d] Download & Run
[z] Download zImage into RAM
[g] Boot linux from RAM
[f] Format the nand flash
[b] Boot the system
[s] Set the boot parameters
[u] Backup NAND Flash to HOST through USB(upload)
[r] Restore NAND Flash from HOST through USB
[q] Goto shell of vivi
Enter your selection:
```

- Σε λίγο θα δούμε το μενού της εικόνας (Εικόνα 4: supervivi menu)

Όπως βλέπουμε υπάρχουν εντολές για διαμόρφωση της NAND [x,f], αναβάθμιση του vivi [v], κατέβασμα καινούριου πυρήνα [k], καθορισμός παραμέτρων εκκίνησης [b], αποθήκευση και επαναφορά της μνήμης flash μέσο USB [u,r], μεταφορά στο κέλυφος του vivi [q] και άλλες που αφορούν συστήματα windows ή χωρίς λειτουργικό.

Προκειμένου να εκτελέσουμε την οποιαδήποτε ενέργεια, επιλέγουμε από το μενού την κατάλληλη επιλογή και ακολουθούμε τυχών οδηγίες εισάγοντας αν χρειαστεί περαιτέρω δεδομένα. Αν απαιτείται μεταφορά αρχείου, μετά την πληκτρολόγηση της επιλογής, εμφανίζεται το μήνυμα “USB host is connected. Waiting a download.”. Μπαίνουμε στο φάκελο που έχουμε αποσυμπιέσει το pushUSB (αν δεν τον έχουμε μετονομάσει λέγεται s3c2410_boot_usb) και πληκτρολογούμε “./s3c2410_boot_usb 'όνομα αρχείου για μεταφορά'”.

4.4 U-boot

Το u-boot είναι ένας boot-loader με πολύ περισσότερες δυνατότητες από το supervivi που μπορεί να εγκατασταθεί σε πολλά ενσωματωμένα συστήματα. Στη συγκεκριμένη περίπτωση, το να αφαιρέσουμε το supervivi είναι λίγο επικίνδυνο. Αν κάτι δεν πάει καλά μπορεί να καταστεί αδύνατο να εκκινήσουμε ξανά την πλακέτα. Βέβαια με τη χρήση H-JTAG μπορούμε να περάσουμε από την αρχή boot-loader, αλλά κάτι τέτοιο θα μπορούσε να αποδειχτεί χρονοβόρο, δεν θα ήταν όμως απαραίτητο.

Το u-boot μπορεί να εγκατασταθεί σα δεύτερος boot-loader στη NAND. Έτσι η πλακέτα εκκινεί το supervivi το οποίο φορτώνει το u-boot από τη NAND και αυτό με τη σειρά του τρέχει το λειτουργικό σύστημα από την κάρτα SD ή μέσω δικτύου. Βέβαια με αυτό τον τρόπο δε μπορούμε να έχουμε

λειτουργικό σύστημα εγκατεστημένο στη μνήμη flush, και επομένως είναι απαραίτητη η χρήση SD κάρτας ή δικτύου. Από την άλλη αυτό δεν είναι κατ' ανάγκη αρνητικό, καθώς η χρήση τέτοιων μέσων έχει πλεονεκτήματα. Πέρα από τον κατά πολύ μεγαλύτερο αποθηκευτικό χώρο, η μνήμη NAND χρησιμοποιείται λιγότερο και επομένως είναι δυσκολότερο να καταστραφεί λόγω υπερβολικής χρήσης. Η μνήμη SD μπορεί επίσης να αντιμετωπίσει πρόβλημα λόγο εκτεταμένης χρήσης, αλλά είναι φθηνότερη και αντικαθίσταται εύκολα. Επιπλέον είναι πιο εύκολο να δοκιμάσουμε διαφορετικά λειτουργικά, απλά αλλάζοντας το αποθηκευτικό μέσο.

Διαδικασία χτισίματος:

Μπορούμε να εγκαταστήσουμε την έκδοση του u-boot που βρίσκετε στη σελίδα της FriendlyARM. Για να έχουμε όμως την τελευταία έκδοση θα προτιμήσουμε το αποθετήριο git όπως και με τον πυρήνα.

- Δημιουργούμε ένα φάκελο, μπαίνουμε και κατεβάζουμε τον κώδικα του u-boot
- mkdir uboot ; cd uboot

git clone git://repo.or.cz/u-boot-openmoko/mini2440.git

- Εξάγουμε τη μεταβλητή export "CROSS_COMPILE='θέση'arm-none-linux-gnueabi-". Μπορούμε βέβαια να παραλείψουμε αυτό το βήμα και να διευκρινίζουμε τη θέση του cross-compiler κάθε φορά που εκτελούμε εντολή που περιλαμβάνει μεταγλώττιση (όπως κάναμε στο χτίσιμο πυρήνα με τη μέθοδο του Bill)
- Μπαίνουμε στο φάκελο mini2440 που έχει δημιουργηθεί και εκτελούμε "make mini2440_config" για να κατασκευαστεί το αρχείο ρυθμίσεων και "make" για να χτιστεί το u-boot.

Υπό φυσιολογικές συνθήκες έχουμε μια εικόνα του u-boot με όνομα "u-boot.bin". Στην έκδοση που χρησιμοποίησα υπήρχαν κάποια προβλήματα, και επομένως ακολουθώντας και πάλι οδηγίες του Bill προστέθηκε ένα βήμα. Αμέσως μετά τη δημιουργία του φακέλου mini2440 με το git, άνοιξα το αρχείο "include/configs/mini2440.h" και έβαλα σε σχόλιο τη γραμμή "#define CONFIG_USE_IRQ 1".

Διαδικασία εγκατάστασης:

- εκκινούμε την πλακέτα, με τον διακόπτη επιλογής εκκίνησης στο NOR και συνδεδεμένα τα καλώδια USB και σειριακό.
- Στο μενού του supervivi, πατάμε το q για να μεταφερθούμε στην κονσόλα του.
- Εκεί πληκτρολογούμε "load ram 0x32000000 <μέγεθος του uboot bin σε bytes> u-boot". Με αυτή την εντολή η πλακέτα θα περιμένει το u-boot σαν είσοδο από τη USB για να το φορτώσει στη θέση μνήμης 0x32000000. Θα δούμε ένα μήνυμα "USB host is connected. Waiting a download. ".
- Μπαίνουμε στο φάκελο του του usbpush και πληκτρολογούμε "./s3c2410_boot_usb <θέση>u-boot.bin" όπου <θέση> η θέση του u-boot.bin. Προσωπικά αντέγραφα το αρχείο προς μεταφορά στο φάκελο του usbpush ώστε να μη χρειάζεται η πληκτρολόγηση θέσης. Μετά τη μεταφορά θα δούμε κάτι όπως:
- csum = 0x9a6e
- send_file: addr = 0x33f80000, len = 0x0003a298
- Error downloading program
- Το "Error downloading program " δεν πρέπει να μας ανησυχήσει.
- Με την εντολή "go 0x32000000" εκτελούμε το πρόγραμμα που μόλις κατεβάσαμε στη μνήμη.
- Τώρα πρέπει να βλέπουμε την προτροπή "MINI24440#" που σημαίνει ότι είμαστε στο περιβάλλον του u-boot. Προετοιμάζουμε τη μνήμη NAND με την εντολή "nand scrub"
- Για να μην αντιμετωπίσουμε προβλήματα σε περίπτωση που υπάρχουν κατεστραμμένα κομμάτια της NAND, δημιουργούμε ένα πίνακα αυτών με την εντολή "nand createbbt". Έτσι το u-boot θα αποφύγει να τα χρησιμοποιήσει.
- Περνάμε το u-boot που εκτελείτε από τη θέση μνήμης 0x32000000 στη NAND "nand write.e 0x32000000 0x0 <μέγεθος του uboot bin σε δεκαεξαδικό> u-boot>"
- Δημιουργούμε κατατμήσεις "dynpart"
- Δημιουργούμε χώρο δυναμικών μεταβλητών "dynenv set u-boot_env"
- Σώνουμε τις μεταβλητές περιβάλλοντος "saveenv"

Σε αυτό το σημείο, αν δεν παρουσιαστούν προβλήματα, μπορούμε να γυρίσουμε το διακόπτη στη μνήμη NAND (αφού απενεργοποιήσουμε την πλακέτα) και να εκκινήσουμε την πλακέτα χρησιμοποιώντας το u-boot σαν boot-loader.

Χρήση του u-boot:

Χρησιμοποίησα το u-boot, για να εκκινήσω λειτουργικό είτε από την κάρτα SD είτε μέσο δικτύου για φάκελο προσβάσιμο μέσο NFS (Network File System, είναι τρόπος δικτύωσης υπολογιστών, κυρίως για Linux).

SD boot:

Η περιγραφή αναφέρεται σε κάρτες με τις εξής τρεις κατατμήσεις:

Κατάτμηση 1: Linux Swap (για εικονική μνήμη)

Κατάτμηση 2: Linux kernel ext2 (ο πυρήνας)

Κατάτμηση 3: Linux file system ext3 (το σύστημα αρχείων μας)

Η πρώτη κατάτμηση μένει βέβαια κενή ώστε να χρησιμοποιηθεί σαν επιπλέον μνήμη. Στη δεύτερη κατάτμηση τοποθετούμε την εικόνα του πυρήνα σε μορφή uImage και με όνομα “uImage”. Η κανονική διαδικασία παραγωγής πυρήνα που έχουμε δει, παράγει πυρήνες σε μορφή zImage οι οποίοι δουλεύουν σε supervivi. Το u-boot όμως απαιτεί ο πυρήνας να είναι σε μορφή uImage. Για να αποκτήσουμε ένα τέτοιο πυρήνα υπάρχουν δύο τρόποι.

1. Είτε να τον παράγουμε από την αρχή, χρησιμοποιώντας την εντολή “CROSS_COMPILE=/usr/local/arm-2008q3/bin/arm-none-linux-gnueabi- ARCH=arm make uImage” αντί για “CROSS_COMPILE=/usr/local/arm-2008q3/bin/arm-none-linux-gnueabi- ARCH=arm make”.
2. Είτε να μετατρέψουμε την εικόνα zImage σε uImage με το εργαλείο mkimage, εκτελώντας “<θέση>mkimage -A arm -O linux -T kernel -C none -a 0x30008000 -e 0x30008000 -d zImage uImage” μέσα στο φάκελο “arch/arm/boot” του πυρήνα. Την τρίτη κατάτμηση τέλος θα την αποσυμπιέσουμε με το σύστημα αρχείων που θα χρησιμοποιήσουμε. Αυτό συνήθως θα είναι σε μορφή tar.gz.

Για να ρυθμίσουμε το u-boot, ενεργοποιούμε την πλακέτα σε NAND και πατάμε ένα πλήκτρο μέσα στα τρία δευτερόλεπτα που μας δίνονται πριν προσπαθήσει το u-boot να εκκινήσει το λειτουργικό σύστημα. Έπειτα περνάμε τις παρακάτω δύο μεταβλητές περιβάλλοντος:

setenv bootcmd mmcinit \; ext2load mmc 0:2 0x31000000 uImage \; bootm 0x31000000

setenv bootargs console=ttySAC0,115200 mini2440=0tb rootfstype=ext3 root=/dev/mmcblk0p3 rw rootwait

και σώνουμε με:

saveenv

Αν επανεκκινήσουμε, την πλακέτα (δίνοντας reset ή πατώντας το πλήκτρο επανεκκίνησης) έχοντας τοποθετημένη την κάρτα SD, θα τρέξει το λειτουργικό μας.

NFS boot:

Με αυτή τη μέθοδο, τόσο ο πυρήνας όσο και το σύστημα αρχείων, δε βρίσκονται σε κάποιο μέσο πάνω στην πλακέτα, αλλά στον κύριο υπολογιστή μας. Τα πλεονεκτήματα της μεθόδου είναι, αφενός η μεγάλη αντοχή του σκληρού δίσκου σε πολλές επανεγγραφές σε σχέση με την κάρτα SD (αυτό είναι ίσως σημαντικό αν η δουλειά που θέλουμε να κάνουμε απαιτεί πολλές αλλαγές σε αρχεία.) αφετέρου, μιας και το σύστημα αρχείων της πλακέτας είναι απλά ένας φάκελος στον υπολογιστή μας, μπορούμε να αλλάζουμε ή να μετακινούμε αρχεία εύκολα, ακόμα και όταν η πλακέτα είναι σε λειτουργία. Αυτό επιταχύνει πολύ τη δουλειά μας σε κάποιες περιπτώσεις, όταν θέλουμε για παράδειγμα να φτιάξουμε ένα πρόγραμμα που να εκτελείτε στην πλακέτα. Αντί να μπαίνουμε στην αργή διαδικασία μεταγλώττισης - μεταφοράς του εκτελέσιμου στην πλακέτα - εκτέλεσης, μεταγλωττίζουμε απευθείας σε φάκελο προσβάσιμο στην πλακέτα.

Βέβαια υπάρχουν και μειονεκτήματα. Καθώς η κάρτα δικτύου χρησιμοποιείται συνεχώς από το u-boot, η συμπεριφορά του δικτύου είναι προβληματική.

Διαδικασία:

- Δημιουργούμε το φάκελο “/export” και μέσα τους φακέλους “kernel” και “fs”. Στο φάκελο kernel τοποθετούμε την εικόνα uImage. Στο φάκελο fs αποσυμπιέσαμε το σύστημα αρχείων.

- Κάνουμε αυτούς τους φακέλους προσβάσιμους μέσω NFS.
- Έπειτα περνάμε στην κάρτα τα παρακάτω ορίσματα, με τον τρόπο που αναφέρετε στη μέθοδο της κάρτας SD.

setenv bootcmd nfs 0x31000000 <Host IP>:/export/kernel/uImage \; bootm
setenv bootargs console=ttySAC0,115200 root=/dev/nfs nfsroot=<Host IP>:/export/fs rw ip=192.168.10.54 mini2440=0tb
ipaddr=<IP πλακέτας>
netmask=<Network mask>
serverip=<Host IP>
saveenv
(όπου <Host IP> ή IP του υπολογιστή που έχουμε τους φακέλους "/export/kernel" και "/export/fs")

4.5 Openembedded

Το openembedded είναι ένα framework για τη δημιουργία διανομών Linux που στοχεύει σε ενσωματωμένα συστήματα, χωρίς να περιορίζεται σε αυτά. Συχνά αναφερόμαστε σε αυτό σαν oe για συντομία. Το σύστημα χτισίματος είναι το BitBake και η λειτουργία του θυμίζει το ebuilds της διανομής Gentoo. Επιλέχθηκε λόγω της ευελιξίας του, και της σχετικά μεγάλης υποστήριξης του από την κοινότητα της mini2440. Η διαδικασία έγινε σε ubuntu Linux (εκδώσεις 9.04, 10.04, 11.10) και επομένως η παρακάτω περιγραφή αφορά κυρίως ubuntu.

Πριν ξεκινήσουμε θα πρέπει να έχουμε εγκατεστημένα τα προγράμματα:
git =>1.6.3.3
bitbake =>1.8.18 (>=1.10.2 για Release-2011.03)
bison
flex
e2fsprogs
m4
curl
ncurses libraries and development headers
zlib development libraries and headers
cvs
subversion
unzip
bzip2 and its development libraries and headers

και ότι άλλο ζητήσει το bitbake την πρώτη φορά που θα το τρέξουμε.

Αφού όλα εγκατασταθούν με επιτυχία, δημιουργούμε ένα φάκελο "oeroot", μπαίνουμε και δημιουργούμε ένα αντίγραφο του git με την εντολή "git clone git://repo.or.cz/openembedded/mini2440.git". Αντί για το κυρίως αποθετήριο του openembedded, χρησιμοποιούμε το προσαρμοσμένο για mini2440. Οι εφαρμογές μπορεί να είναι λιγότερο ενημερωμένες, αλλά έχουμε λιγότερο κίνδυνο να αντιμετωπίσουμε προβλήματα σχετικά με το υλικό.

Μετά από επιτυχή εκτέλεση της εντολής, βλέπουμε ένα φάκελο mini2440 μέσα στον oeroot. Αυτός ο φάκελος περιέχει τα αρχεία που θα χρειαστεί το bitbake για να κατεβάσει και να χτίσει τα αρχεία μας.

Δημιουργούμε τους φακέλους "build" και "sources" μέσα στον oeroot. Στο φάκελο sources θα μπει ο πηγαίος κώδικας των αρχείων που χρειαζόμαστε και στο build θα τοποθετηθούν τα έτοιμα αρχεία.

Έπειτα δημιουργούμε ένα φάκελο conf μέσα στο build (mkdir build/conf) και αντιγράφουμε το αρχείο mini2440_local_conf_example.conf που βρίσκετε στο φάκελο mini2440 ως "local.conf" (cp mini2440/mini2440_local_conf_example.conf build/conf/local.conf). Αυτό το αρχείο περιέχει τις ενδεικτικές ρυθμίσεις για το bitbake. Για να αλλάξουμε αυτές τις ρυθμίσεις ανοίγουμε το local.conf και αλλάζουμε ότι χρειάζεται. Στην εν λόγω περίπτωση:

DL_DIR = <διαδρομή στο>/sources
BBFILES = <διαδρομή στο>/recipes/*/*.bb

TMPDIR = <διαδρομή στο>build (πρέπει να προσέξουμε να μην έχει / στο τέλος)

ASSUME_PROVIDED είναι τα προγράμματα που θεωρούμε δεδομένο ότι υπάρχουν, αν κάτι δεν υπάρχει και ξέρουμε ότι δε χρειάζεται μπορούμε να το σβήσουμε.

Τέλος, σβήνουμε την τελευταία γραμμή του αρχείου "# REMOVE_THIS_LINE....". Η γραμμή αυτή λειτουργεί σαν προστασία, για να μην τρέξει κάποιος το bitbake χωρίς να διαβάσει το αρχείο ρυθμίσεων.

Φυσικά μπορούμε να κάνουμε και άλλες αλλαγές, για να παραμετροποιήσουμε το σύστημα αρχείων που θα δημιουργηθεί.

Έπειτα εξάγουμε τη μεταβλητή "export BBPATH=<διαδρομή στο>mini2440:<διαδρομή στο>build".

Αν τρέχουμε ubuntu πρέπει και να ρυθμίσουμε το "/bin/sh" να δίνει στο "/bin/bash". Για να το κάνουμε πληκτρολογούμε "sudo dpkg-reconfigure dash" και επιλέγουμε "NO" όταν εμφανιστεί η επιλογή.

Τώρα εκτελούμε "bitbake -v task-base" μέσα στο φάκελο build. Αυτή η εντολή θα χτίσει όλα τα βασικά εργαλεία που θα χρειαστούμε όπως cross-compiler, βιβλιοθήκες και συνταγές τόσο για ολόκληρες εικόνες συστήματος αρχείων όσο και μεμονωμένων εφαρμογών. Μερικές φορές το bitbake μπορεί να βγάλει ένα λάθος σχετικό με το αρχείο /proc/sys/vm/mmap_min_addr. Αν συμβεί κάτι τέτοιο, θέτουμε την τιμή του αρχείου ίση με 0 "echo 0 > /proc/sys/vm/mmap_min_addr" και εκτελούμε και πάλι το bitbake.

Τώρα μπορούμε να χτίσουμε συνταγές, δημιουργώντας τα πακέτα ή τις εικόνες που χρειαζόμαστε. Ως παράδειγμα ξεκινάμε τρέχοντας "bitbake -v console-image". Αυτό δημιουργεί ένα σύστημα αρχείων με τα απολύτως βασικά και βέβαια χωρίς γραφικό περιβάλλον. Τα έτοιμα αρχεία θα τα βρούμε στο φάκελο "tmp/deploy/glibc/images/mini2440". Η εικόνα που παράγεται είναι σε διάφορες μορφές, οπότε διαλέγουμε αυτή που μας βολεύει (tar.gz συνήθως).

Το bitbake εκτός από το σύστημα αρχείων μπορεί να μας δώσει επίσης πυρήνα και cross-compiler. Δεν τα χρειαζόμαστε όμως γιατί έχουμε ήδη. Στο διαδίκτυο βέβαια κάποιος ισχυρίστηκε ότι ο cross-compiler που παράγει το bitbake είναι καλύτερος, και πως ο επίσημος της FriendlyARM του παρουσίασε κάποια προβλήματα. Εγώ ωστόσο δεν αντιμετώπισα κάποιο πρόβλημα με κανέναν από τους δύο.

4.6 Angstrom

Η έκδοση του openembedded για τη mini2440 (όπως και για πολλές άλλες συσκευές) συντηρείται από τη διανομή angstrom. Μετά την εγκατάσταση μπορούμε να ρυθμίσουμε χειροκίνητα την κάρτα δικτύου (υπάρχει πρόβλημα με την αυτόματη αναγνώριση δικτύου), να συνδεθούμε στο διαδίκτυο και να κάνουμε αυτοματοποιημένη εγκατάσταση προγραμμάτων χρησιμοποιώντας το διαχειριστή πακέτων opkg. Επίσης μπορούμε να τον χρησιμοποιήσουμε για να εγκαταστήσουμε πακέτα που χτίσαμε από συνταγές του bitbake ή που κατεβάσαμε από τη σελίδα της διανομής (http://www.angstrom-distribution.org/).

Επίσης στη σελίδα (http://narcissus.angstrom-distribution.org/) θα βρούμε ένα αυτοματοποιημένο πρόγραμμα, που μπορεί να δημιουργήσει για εμάς εικόνες κατά παραγγελία. Έτσι μπορούμε να αποφύγουμε την παραπάνω διαδικασία, εξοικονομώντας χρόνο. Οι εικόνες που παράγει αυτός ο τρόπος (narcissus) είναι λειτουργικές και δυνατόν να παραμετροποιηθούν. Υπάρχουν αρκετές επιλογές τόσο για τις ρυθμίσεις του λειτουργικού, όσο και για τις προεγκατεστημένες εφαρμογές. Βέβαια με το bitbake, αφού έχουμε στον υπολογιστή μας τον κώδικα των εφαρμογών, έχουμε μεγαλύτερες δυνατότητες παραμετροποίησης ή επίλυσης προβλημάτων.

Πρόβλημα του openembedded αποτελεί η υποστήριξη γραφικού περιβάλλοντος. Πολλά μέλη της κοινότητας ψάχνουν λύση στο θέμα, ωστόσο δεν βρήκα μια ικανοποιητική λύση. Το καλύτερο που κατάφερα να κάνω ήταν να τρέξω τη βαθμονόμηση οθόνης. Η έλλειψη αυτή δεν επηρεάζει βέβαια τους βασικούς στόχους της εργασίας, θα έκανε όμως την πλακέτα πιο αυτόνομη και πλήρη. Το πρόβλημα παραμένει και με τις εικόνες που παράγουμε με το bitbake και με τις εικόνες του narcissus.

4.7 Web Server

Η παρουσία διακομιστή ιστοσελίδων κρίθηκε απαραίτητη, ώστε να κατασκευαστεί μια διεπαφή ελέγχου και παρακολούθησης. Οι διεπαφές φυλλομετρητή μπορούν να είναι εύχρηστες, ευπαρουσίαστες και προσβάσιμες από οπουδήποτε. Επιπλέον, θα μπορούσαν να τρέξουν και στην ίδια την πλακέτα, αφού και αυτή μπορεί να τρέχει το δικό της φυλλομετρητή. Ο διακομιστής ιστοσελίδων που θα χρησιμοποιήσουμε, θα πρέπει να έχει βέβαια πολύ μικρές απαιτήσεις συστήματος, ώστε να τρέξει χωρίς πρόβλημα στην πλακέτα.

4.7.1 Boa Server

Ο προεγκατεστημένος web server στη διανομή της FriendlyARM είναι ο Boa server. Είναι ένας ελαφρύς διακομιστής κατάλληλος για ενσωματωμένα συστήματα. Η προσπάθεια βέβαια για μικρές απαιτήσεις συστήματος έχει αντίκτυπο στις δυνατότητες του, οι οποίες είναι πολύ περιορισμένες. Ίσως να ήταν αρκετός για την παρούσα εργασία, αλλά δεδομένου ότι στο openembedded δεν είναι προεγκατεστημένος προτίμησα κάτι πιο πλήρες.

4.7.2 Cherokee Server

Ο cherokee είναι επίσης ένας ελαφρύς διακομιστής ιστοσελίδων. Είναι βέβαια βαρύτερος από τον boa. Έχει όμως σημαντικά περισσότερες δυνατότητες, τόσο που η χρήση του δεν περιορίζεται σε ενσωματωμένες συσκευές. Υπάρχει διαθέσιμος σχεδόν για όλα τα λειτουργικά συστήματα και με πολλές υποστηριζόμενες γλώσσες.

Υποστηρίζεται από το openembedded και η εγκατάσταση του σε αυτό είναι εύκολη. Παρόλα αυτά λόγω του παλιού σετ εντολών του επεξεργαστή της πλακέτας, η διαθέσιμη έκδοση είναι απαρχαιωμένη. Έτσι δε βρήκα τρόπο να ενεργοποιήσω την υποστήριξη php, κάτι που στις κανονικές εκδόσεις είναι πολύ εύκολο. Τελικά η διεπαφή γράφηκε με C cgi. Αυτό δεν είναι κακό, γιατί και το πρόγραμμα μετρήσεων είναι γραμμένο στη C. Έτσι υπάρχει μια ομοιομορφία στον κώδικα.

4.8 SysFS

Το sysfs (SYStem File System) (σύστημα αρχείων συστήματος) είναι ένας κομψός και εύκολος τρόπος να αποκτήσουμε πρόσβαση σε κάποιες συσκευές. Πρόκειται για ένα εικονικό σύστημα αρχείων (virtual file system) το οποίο: εξάγει πληροφορίες, λειτουργίες του πυρήνα, των οδηγών (drivers) συσκευών με τη μορφή αρχείων και των φακέλων, τα οποία γίνονται προσβάσιμα χωρίς τη χρήση ειδικών βιβλιοθηκών.

Εισήχθη στον πυρήνα του linux από την έκδοση 2.6 (υπήρχε και στη 2.5 η οποία όμως ήταν 2.6 σε στάδιο ανάπτυξης). Σκοπός του ήταν να λύσει προβλήματα των παλαιότερων εκδόσεων όπως η έλλειψη δυνατότητας τοποθέτησης και άμεσης λειτουργίας και να προσφέρει ένα σωστά δομημένο τρόπο πρόσβασης στους οδηγούς συσκευών.

Κατά την εκκίνηση του συστήματος το sysfs προσαρτάτε στο φάκελο “/sys”. Όλα τα περιεχόμενα του φακέλου υπάρχουν μόνο στη μνήμη του υπολογιστή και όχι σε κάποιο μη πτητικό μέσο αποθήκευσης. Παράγονται από τον πυρήνα. Κάθε αλλαγή που κάνουμε στα αρχεία περνάει σε αυτόν για να προβεί στις απαραίτητες ενέργειες, ενώ τα αρχεία ενημερώνονται σε πραγματικό χρόνο για ό,τι αλλάξει. Η λειτουργία του γίνεται καλύτερα αντιληπτή στην ενότητα GPIO όπου το βλέπουμε σε χρήση.

4.9 GPIO

Μερικές φορές, (όπως τώρα), είναι χρήσιμες μερικές ψηφιακά ελεγχόμενες θύρες εισόδου εξόδου που να μπορούμε να τις ελέγξουμε σχετικά εύκολα και να μπορούμε να συνδέσουμε πάνω τους οτιδήποτε. Αυτές είναι οι GPIO (General Purpose Input/Output) (γενικού σκοπού εισόδου/εξόδου). Όπως μαρτυρά το όνομα τους, είναι ακροδέκτες γενικού σκοπού που από προεπιλογή είναι απενεργοποιημένοι. Μπορούμε να τα ενεργοποιήσουμε και να τα ελέγξουμε μέσο λογισμικού. Έχουμε τη δυνατότητα να ορίσουμε τη φορά τους (είσοδος – έξοδος) να διαβάσουμε την τιμή τους και να την αλλάξουμε.

Ένας τρόπος χειρισμού τους είναι με τη χρήση βιβλιοθηκών όπως η “gpio.h”. Σε αυτή την

περίπτωση το πρόγραμμά μας έχει απευθείας πρόσβαση στο υλικό μέσο του πυρήνα.

Ο πιο εύκολος τρόπος όμως (όταν πρόκειται τουλάχιστο για καινούρια συστήματα) είναι η χρήση του εικονικού συστήματος αρχείων sysfs (4.8 SysFS). Όταν ο πυρήνας μας έχει ενεργοποιημένο το sysfs και τους drivers του GPIO, δημιουργείτε ο φάκελος "/sys/class/gpio". Μέσα σε αυτό το φάκελο υπάρχει ένας φάκελος gpiochipN για κάθε chip gpio της συσκευής, καθώς και δύο αρχεία, τα export και unexport. Προκειμένου να χρησιμοποιήσουμε κάποιο ακροδέκτη δημιουργούμε ένα φάκελο gpioN για αυτόν χρησιμοποιώντας το export. Για να διαβάσουμε, να γράψουμε δεδομένα ή να αλλάξουμε την κατεύθυνση του ακροδέκτη, απλά αλλάζουμε της τιμές των αρχείων (direction και value) που περιέχει. Με το unexport μπορούμε να διαγράψουμε ένα φάκελο ακροδέκτη αν δεν τον χρειαζόμαστε πλέον.

Αναλυτικά. Ο εν λόγο φάκελος στην mini2440 αρχικά περιέχει τα παρακάτω αρχεία:

```
root@mini2440:/sys/class/gpio# ls
export       gpiochip128  gpiochip192  gpiochip32   gpiochip96
gpiochip0    gpiochip160  gpiochip224  gpiochip64   unexport
```

Άρα υπάρχουν οκτώ gpio chip στη συσκευή. Κάποια από αυτά συνδέονται στους 34 ακροδέκτες γενικού σκοπού ενώ άλλα χρησιμοποιούνται για τα ενσωματωμένα LED, τους διακόπτες γενικής χρήσης και πιθανό άλλες συσκευές της πλακέτας. Μπορούμε να βρούμε ποιος φάκελος αντιστοιχεί σε κάθε chip διαβάζοντας το αρχείο label του κάθε φακέλου. Για να δούμε για παράδειγμα σε ποιο chip αντιστοιχεί ο φάκελος gpiochip32 θα πληκτρολογήσουμε στο τερματικό:

```
root@mini2440:/sys/class/gpio#                                cat
/sys/class/gpio/gpiochip32/label
GPIOB
```

Επομένως ο φάκελος gpiochip32 του sysfs αντιστοιχεί στο chip GPIOB του εγχειριδίου της πλακέτας.

Για να δούμε την αντίστοιχη πληροφορία για όλα τα chip, ενώ είμαστε στο φάκελο "/sys/class/gpio" εκτελούμε το παρακάτω:

```
root@mini2440:/sys/class/gpio# for i in gpiochip* ; do echo `cat
$i/label`: `cat $i/base` ; done
GPIOA: 0
GPIOE: 128
GPIOF: 160
GPIOG: 192
GPIOH: 224
GPIOB: 32
GPIOC: 64
GPIOD: 96
```

Τώρα πρέπει να ξέρουμε, με βάση το εγχειρίδιο "MINI2440_USER_MANUAL.pdf", ποιον ακροδέκτη χρειαζόμαστε. Ας θεωρήσουμε ότι θέλουμε να χειριστούμε το δεύτερο LED. Με βάση τη σελίδα 231 του εγχειριδίου είναι το GPB6. Άρα θέλουμε τον έκτο ακροδέκτη του GPIOB. Το GPIOB αντιστοιχεί στο φάκελο gpiochip32. Ο ακροδέκτης που ζητάμε λοιπών είναι ο 32 + 6 = 38. Για να δημιουργήσουμε φάκελο για αυτόν απλά τυπώνουμε την τιμή του στο αρχείο export:

```
root@mini2440:/sys/class/gpio# echo 38 > /sys/class/gpio/export
root@mini2440:/sys/class/gpio# ls
export        gpiochip0     gpiochip160   gpiochip224   gpiochip64
unexport
gpio38       gpiochip128  gpiochip192  gpiochip32   gpiochip96
```

Βλέπουμε ότι δημιουργήθηκε ο φάκελος gpio38. Μέσα στον οποίο υπάρχουν τα αρχεία:

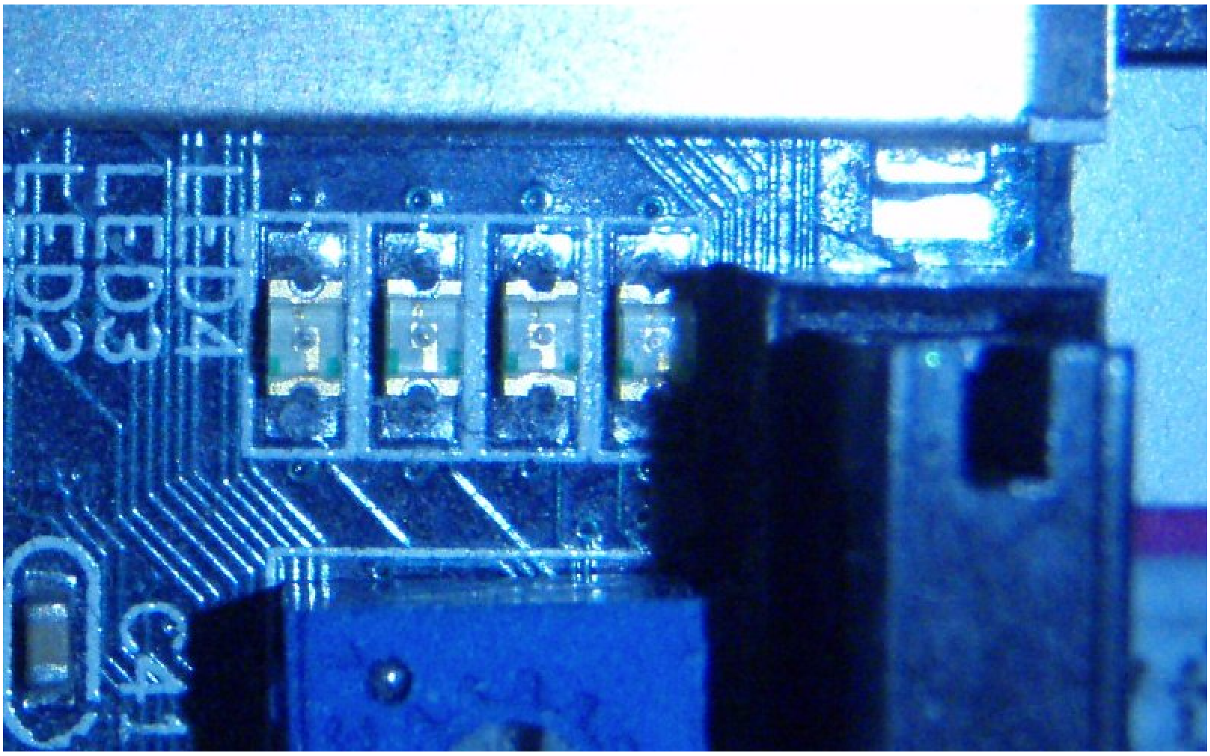
```
root@mini2440:/sys/devices/virtual/gpio/gpio38# ls
direction  power      subsystem  uevent     value
```

Μας ενδιαφέρουν, σε αυτή την περίπτωση τουλάχιστον, τα direction και value. Όπως καταλαβαίνουμε από το όνομα τους, το direction συμβολίζει την κατεύθυνση του ακροδέκτη (in – out) ενώ το value την τιμή του (0 -1).

Με μία γρήγορη επισκόπηση βλέπουμε της προεπιλεγμένες τιμές:

```
root@mini2440:/sys/devices/virtual/gpio/gpio38# cat direction
```

```
in
root@mini2440:/sys/devices/virtual/gpio/gpio38# cat value
1
```



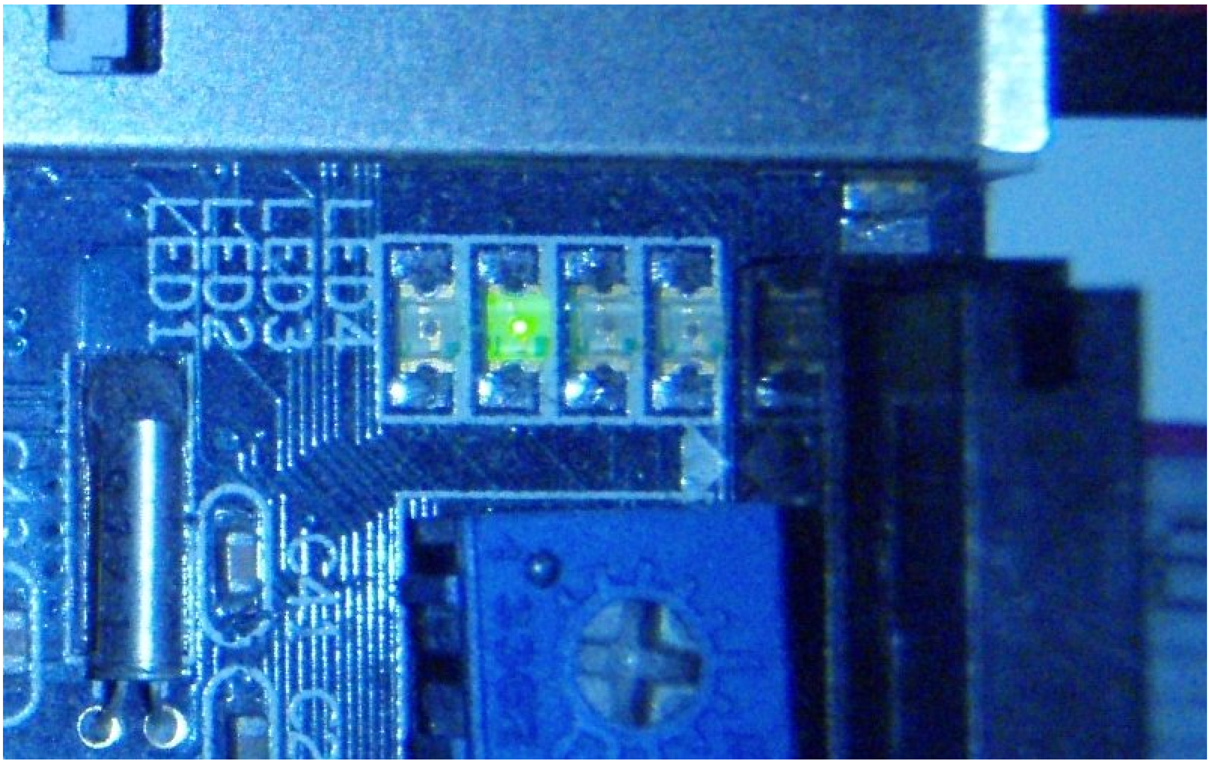
Εικόνα 5: LED σβηστό

Αρχικά ο ακροδέκτης είναι δηλωμένος σαν είσοδος. Στη φωτογραφία LED01 βλέπουμε όλα τα LED σβηστά (Εικόνα 5: LED σβηστό).

Αλλάζοντας λίγο τις τιμές:

```
root@mini2440:/sys/devices/virtual/gpio/gpio38# echo out > direction
root@mini2440:/sys/devices/virtual/gpio/gpio38# cat value
0
```

Αλλάζοντας την κατεύθυνση σε έξοδο η τιμή αλλάζει αυτόματα σε 1 και το LED ανάβει. Το αποτέλεσμα φαίνεται στη φωτογραφία (Εικόνα 6: LED αναμμένο).



Αν θέλουμε να καταργήσουμε το φάκελο η εντολή που θα πληκτρολογήσουμε είναι “echo 38 > /sys/class/gpio/unexport”. Πρέπει να έχουμε υπόψιν μας ότι με κάθε νέα εκκίνηση του συστήματος οι φάκελοι αυτοί εξαφανίζονται, οπότε η διαδικασία πρέπει να επαναλαμβάνετε (χειροκίνητα ή αυτοματοποιημένα) κάθε φορά που εκκινούμε.

Σημείωση: Όταν η πλακέτα τρέχει Angstrom openembedded, το LED1 αναβοσβήνει συνεχώς ως ένδειξη λειτουργίας, ενώ το LED3, κυρίως κατά την εκκίνηση της συσκευής, αναβοσβήνει σα LED σκληρού δίσκου. Όπως είδαμε πιο πάνω δεν υπάρχουν φάκελοι για αυτά στο sysfs οπότε πρέπει να τα προγραμμάτισαν με χρήση βιβλιοθηκών. Για να αποφύγουμε πιθανές παρεμβολές θα χρησιμοποιήσουμε τα LED 2 και 4 που είναι ανενεργά. Οι φωτογραφίες τραβήχτηκαν τη στιγμή που το LED1 ήταν σβηστό για να φαίνεται καλύτερα η κατάσταση του LED2 που χρησιμοποιήσαμε παραπάνω. Από εδώ και στο εξής τα LED 2 και 4 θα ονομάζονται συχνά LED1 και 2 αντίστοιχα.

4.10 SQLite3

Για την αποθήκευση των δεδομένων που συλλέγονται από τον αισθητήρα έκρινα καλύτερη τη χρήση της sqlite3. Δεν έχει μεγάλες απαιτήσεις σε επεξεργαστική ισχύ και είναι πολύ λειτουργική. Οι παραδοσιακές προτάσεις SQL όπως η MySQL δίνουν περισσότερες δυνατότητες, αλλά και πάλι οι δυνατότητες της είναι πολύ περισσότερες από τις απαιτήσεις της εργασίας. Είναι προσβάσιμη με πολλούς τρόπους, μέσω γραμμής εντολών και πολλών γλωσσών προγραμματισμού. Πλέον είναι ενσωματωμένη στην php (>=5.3). Το κυριότερο σε αυτή την περίπτωση, υποστηρίζεται άψογα από τη C.

Εγκατάσταση και μεταγλώττιση:

Τα πακέτα που χρησιμοποίησα για την εγκατάσταση τις sqlite3 στην πλακέτα ήταν τα (libc6_2.6.1-r15.1_armv4t.ipk, libsqlite3-0_3.6.5-r0.1_armv4t.ipk και sqlite3_3.6.5-r0.1_armv4t.ipk). Για να τη χειριστούμε μέσω της C χρειαζόμαστε τη βιβλιοθήκη sqlite3.h που βρίσκετε στο πακέτο (libsqlite3-dev)

Προκειμένου να μεταγλωττίσουμε ένα πρόγραμμα της C που χρησιμοποιεί SQLite, πρέπει να πρέπει καταρχήν να προσθέσουμε κάποια αρχεία στους φακέλους του μεταγλωττιστή. Στο φάκελο “arm-2008q3/lib/gcc/arm-none-linux-gnueabi/4.3.2/include” προσθέτουμε το sqlite3.h. Στο φάκελο “arm-2008q3/arm-none-linux-gnueabi/libc/lib” το libsqlite3.so.<έκδοση> (στην περίπτωση μας

libsqlite3.so.0.8.6) και κάνουμε τις συντομεύσεις “libsqlite3.so.0” και “libsqlite3.so” να δείχνουν σε αυτό. Στον κώδικα του προγράμματος πρέπει να προσθέσουμε τη βιβλιοθήκη “sqlite3.h” (#include <sqlite3.h>). Τέλος, δίνοντας την εντολή για τη μεταγλώττιση, πρέπει να προσθέσουμε την παράμετρο lsqlite3 (gcc <το πρόγραμμα μας.c> -lsqlite3).

4.11 SPI

Το SPI είναι ένα ελαστικό πρωτόκολλο μεταφοράς δεδομένων (οι προδιαγραφές του δεν καθορίζονται με πλήρη ακρίβεια) που τυποποιήθηκε από τη Motorola. Ολογράφως λέγετε Serial **Peripheral Interface Bus** (δίαυλος σειριακής περιφερειακής διεπαφής). Μερικές φορές το SPI αναφέρετε ως σειριακός δίαυλος τεσσάρων καλωδίων (Αν και χρησιμοποιείται και έκδοση με τρία καλώδια).

Ακολουθεί το μοντέλο αφέντη-σκλάβου (master-slave) και ένας αφέντης είναι δυνατό να χειριστεί πολλούς σκλάβους.

Τα τέσσερα καλώδια που απαιτούνται είναι:

SCLK (Serial Clock): Παλμός ρολογιού που δίνετε από τον αφέντη και χρονίζει την επικοινωνία.

MOSI, SIMO (Master Output, Slave Input): (έξοδος αφέντη, είσοδος σκλάβου) μεταβιβάζει την εντολή από τον αφέντη στο σκλάβο.

MISO, SOMI (Master Input, Slave Output): (είσοδος αφέντη, έξοδος σκλάβου) μεταβιβάζει την απάντηση τού σκλάβου στον αφέντη.

SS (Slave Select): Επιλογή σκλάβου, είναι έξοδος από τον αφέντη και συνήθως ενεργοποιεί το σκλάβο όταν η έξοδος είναι 0.

Ακριβώς επειδή το πρωτόκολλο είναι ελαστικό τα ονόματα αυτά μπορεί να ποικίλουν! Έτσι το κάθε καλώδιο εμφανίζετε και ως εξής:

SCLK (Serial Clock): SCK, CLK

MOSI, SIMO (Master Output, Slave Input): SDI, DI, DIN, SI

MISO, SOMI (Master Input, Slave Output): SDO, DO, DOUT, SO

SS (Slave Select): nCS, CS, CSB, CSN, nSS, STE

Οι ονομασίες καθορίζονται από την οπτική γωνία. Στο εγχειρίδιο της πλακέτας (η οποία μπορεί να χρησιμοποιηθεί μόνο σαν αφέντης) οι ονομασίες είναι MOSI, MISO, CLK, nSS. Στο εγχειρίδιο του αισθητήρα (που μπορεί να είναι μόνο σκλάβος) ονομάζονται SDI, SDO, SCK, CS.

SS: Σε περίπτωση που έχουμε μόνο ένα σκλάβο, αλλάζει τιμή για να τον ενεργοποιήσει. Άλλοι ενεργοποιούνται σε υψηλή και άλλοι σε χαμηλή τάση. Αν έχουμε περισσότερους από ένα σκλάβους συνδεδεμένους στον ίδιο αφέντη, πρέπει να έχουμε ένα SS στον καθένα, που θα παίρνει χαμηλή (ή υψηλή) τιμή την ώρα της επικοινωνίας και θα κρατάει υψηλή (ή χαμηλή αντίστοιχα) σε κάθε άλλη περίπτωση. Έτσι μπορούμε σε ένα αφέντη να συνδέσουμε όσους σκλάβους θέλουμε, χρησιμοποιώντας τα ίδια καλώδια επικοινωνίας (SCLK, MOSI, MISO) συνδεδεμένα σε σειρά. Χρειαζόμαστε απλά ένα ξεχωριστό SS για κάθε σκλάβο.

SCLK: Η όλη επικοινωνία συντονίζετε από ένα παλμό ρολογιού που στέλνετε από τον αφέντη μέσω του καλωδίου SCLK.

MOSI: Αφού ενεργοποιήσουμε το σκλάβο με το SS. Λαμβάνοντας υπόψιν τον παλμό του SCLK, το MOSI καλώδιο μεταφέρει δεδομένα στον σκλάβο.

MISO: Αφού ο σκλάβος παραλάβει τα δεδομένα από τον αφέντη, και εν το SS συνεχίζει να είναι ενεργοποιημένο, ο σκλάβος θα στείλει την απάντηση μέσο του MISO.

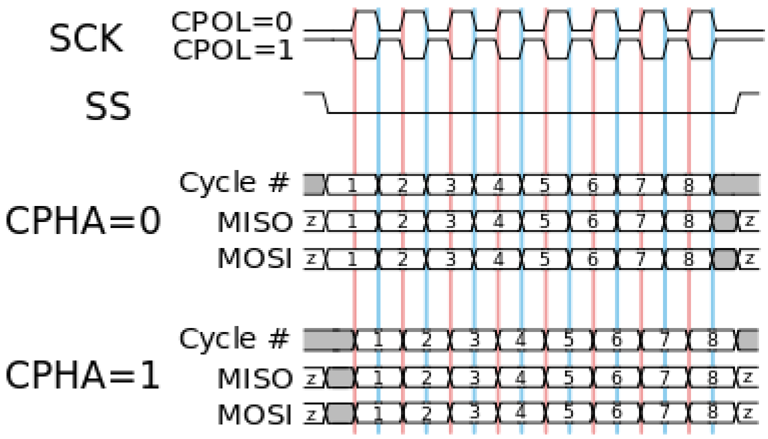
MODS: Όπως αναφέραμε παραπάνω, το SPI είναι ελαστικό πρωτόκολλο, δηλαδή δεν είναι αυστηρά καθορισμένο και δίνει δυνατότητες παραμετροποίησης. Πέρα από τη συχνότητα του παλμού και την καθυστέρηση ανάμεσα σε δύο διαδοχικές εντολές (delay), μπορούμε να ρυθμίσουμε πότε ξεκινάει η αποστολή και η ανάγνωση στοιχείων σε σχέση με τον παλμό, σε ποια κατάσταση του SS και άλλα.

Πιο συγκεκριμένα, κάποιες από τις επιλογές που έχουμε είναι:

- Η συχνότητα του παλμού που καθορίζει και την ταχύτητα μεταφοράς δεδομένων
- Η καθυστέρηση (delay) μεταξύ δύο εντολών
- Το μέγεθος της λέξης σε bit

- Αν το πιο σημαντικό ή το λιγότερο σημαντικό bit θα προηγείται.
- Αν η συσκευή θα δουλεύει σε λειτουργία loop
- Αν ο σκλάβος ενεργοποιείται στο υψηλό ή το χαμηλό σήμα
- Αν θα ενεργοποιήσουμε τη λειτουργία τριών καλωδίων (επικοινωνία με ένα κανάλι)

Σε συντομία όταν μιλάμε για spi mods, εννοούμε το πότε στέλνουμε και πότε λαμβάνουμε δεδομένα, με βάση την πολικότητα του παλμού ρολογιού και τη φάση του παλμού δεδομένων. Συμβολίζουμε τον παλμό ρολογιού με CPOL και τον παλμό δεδομένων με CPHA. Για την απεικόνιση τους, συχνά χρησιμοποιούνται δύο bit, το πιο σημαντικό για το CPOL και το λιγότερο σημαντικό για το CPHA. Η τιμή 1 για το CPOL σημαίνει ότι η βασική τιμή του ρολογιού είναι 1 (κάθε παλμός είναι πτώση στο 0) ενώ η τιμή 0 είναι το αντίστροφο. Για το CPHA ο ρόλος αλλάζει ανάλογα με το CPOL. Έτσι το CPOL 0 και CPHA 0 σημαίνει ότι τα δεδομένα λαμβάνονται κατά την ακμή του παλμού, ενώ στέλνετε κατά τη πτώση του, αντίστροφα για τιμή 1. τα πράγματα αντιστρέφονται για CPOL 1. Το σχήμα (Εικόνα 7: SPI mods) βοηθάει να κατανοήσουμε τα παραπάνω.



Εικόνα 7: SPI mods

Τα 4 mod που προκύπτουν αριθμούνται με βάση τον παρακάτω πίνακα (Πίνακας 1: SPI mods 2bit).

Mode	CPOL	CPHA
0	0	0
1	0	1
2	1	0
3	1	1

Πίνακας 1: SPI mods 2bit

Αναλυτικά:

- Mod 0: Βασική τιμή ρολογιού 0, τα δεδομένα λαμβάνονται με την ακμή του παλμού και στέλνονται με την πτώση του.
- Mod 1: Βασική τιμή ρολογιού 0, τα δεδομένα λαμβάνονται με την πτώση του παλμού και στέλνονται με την ακμή του.
- Mod 2: Βασική τιμή ρολογιού 1, τα δεδομένα λαμβάνονται με την πτώση του παλμού και στέλνονται με την ακμή του .
- Mod 3: Βασική τιμή ρολογιού 1, τα δεδομένα λαμβάνονται με την ακμή του παλμού και στέλνονται με την πτώση του.

Πρέπει να σημειωθεί πως συχνά οι κατασκευαστές δε μας λένε ποιο είναι το spi mod της συσκευής μας με νούμερο, αλλά περιγράφοντάς το με βάση αυτά τα χαρακτηριστικά. Συνήθως με τη βοήθεια κάποιου σχήματος.

4.12 SPI στην περίπτωση μας

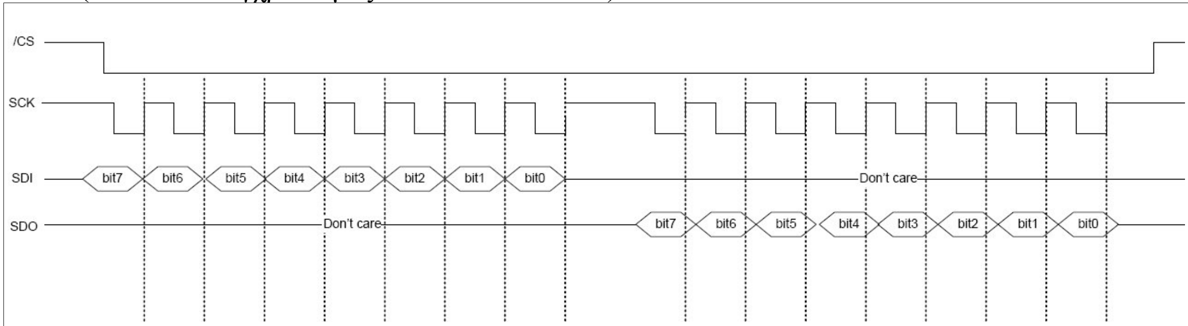
Για να χειριστούμε τον τον αισθητήρα θερμότητας και υγρασίας DC-SS500 (3.3.2 Αισθητήρας) μέσο SPI, χρειάζεται απλά να στείλουμε ένα byte δεδομένων σαν εντολή και να παραλάβουμε ένα σαν απάντηση. Τόσο οι ερωτήσεις όσο και οι απαντήσεις είναι byte integer και απεικονίζονται συνήθως σε δεκαεξαδική μορφή.

Υποστηρίζονται τέσσερις πιθανές εντολές.

0xa0	Εμφανίζει την τρέχουσα σχετική υγρασία σε (%)
0xb0	Εμφανίζει την τρέχουσα θερμοκρασία σε βαθμούς Κελσίου °C
0xb1	Εμφανίζει την τρέχουσα θερμοκρασία σε βαθμούς Φαρενάιτ °F
0xc0	Επιστρέφει 0xAF όταν το σύστημα είναι απασχολημένο. Για να μάθουμε την κατάσταση του συστήματος.

Πίνακας 2: Εντολές DC-SS500

Προκειμένου να καταλάβουμε το υποστηριζόμενο mod, στο εγχειρίδιο του αισθητήρα υπήρχε η εικόνα (Εικόνα 8: συγχρονισμός SPI του DC-SS500).



Εικόνα 8: συγχρονισμός SPI του DC-SS500

Από το σχήμα βλέπουμε ότι η βασική τιμή ρολογιού είναι 1. Άρα αποκλείονται τα mod 0 και 1. Επίσης τα δεδομένα διαβάζονται στην ακμή του παλμού, άρα έχουμε mod3. Επίσης βλέπουμε ότι το πιο σημαντικό bit πάει πρώτο και ότι το CS (επιλογή σκλάβου) πρέπει να είναι 0 κατά τη διάρκεια της επικοινωνίας και 0 σε κάθε άλλη περίπτωση.

Επιπλέον μας δίνονται, η συχνότητα ρολογιού που πρέπει να είναι μεταξύ 3kHz και 2MMz, και η καθυστέρηση (delay) μεταξύ δύο διαδοχικών εντολών που πρέπει να είναι πάνω από 0.2 δευτερόλεπτα.

Κεφάλαιο - 5 Υλοποίηση

Μετά από όσα αναλύονται στο κεφάλαιο 4, έχουμε στα χέρια μας, μια πλακέτα με εγκατεστημένο λειτουργικό σύστημα και εφαρμογές. Έχοντας κάνει και κάποιες δοκιμές και πειράματα, έχουμε αποκομίσει αρκετή γνώση για να σχεδιάσουμε και να υλοποιήσουμε την εφαρμογή. Όπως έχει ειπωθεί και σε προηγούμενο κεφάλαιο, τα προγράμματα θα γραφούν στη C, είτε πρόκειται για εκτελέσιμα είτε για cgi που υλοποιούν τη διεπαφή. Για βάση δεδομένων θα χρησιμοποιηθεί η SQLite3 και ως webserver ο cherokee.

5.1.1 Δυνατότητες

Η τελική υλοποίηση θα χωρίζεται σε διεπαφή και παρασκήνιο. Στο παρασκήνιο θα γίνεται ανάγνωση δεδομένων από τον αισθητήρα και αποθήκευση των μετρήσεων. Επιπλέον τα LED 2 και 4 (που θα ονομάζονται 1 και 2 αντίστοιχα (Σημείωση)) θα παίζουν το ρόλο ανάδρασης. Στη θέση τους θα μπορούσαν να βρίσκονται συσκευές όπως συστήματα θέρμανσης, ψύξης ή εξαερισμού. Η εφαρμογή παρασκηνίου, θα αναβοσβήνει τα LED (ενεργοποιεί απενεργοποιεί συσκευές), όταν οι τιμές που διαβάζει από τον αισθητήρα, ξεφεύγουν από κάποια όρια που ορίζουμε μέσω της διεπαφής.

Η διεπαφή θα μας δίνει τη δυνατότητα, μέσα από το φυλλομετρητή μας, να επιβλέπουμε τις τρέχουσες τιμές θερμοκρασίας και υγρασίας, τις μετρήσεις που έχουν καταγραφή. Θα παρέχει επίσης τρόπο επίβλεψης και χειρισμού των LED, τόσο χειροκίνητα, όσο και αυτόματα. Ο αυτόματος τρόπος είναι η ρύθμιση ορίων και η εναπόθεση του χειρισμού στο παρασκήνιο.

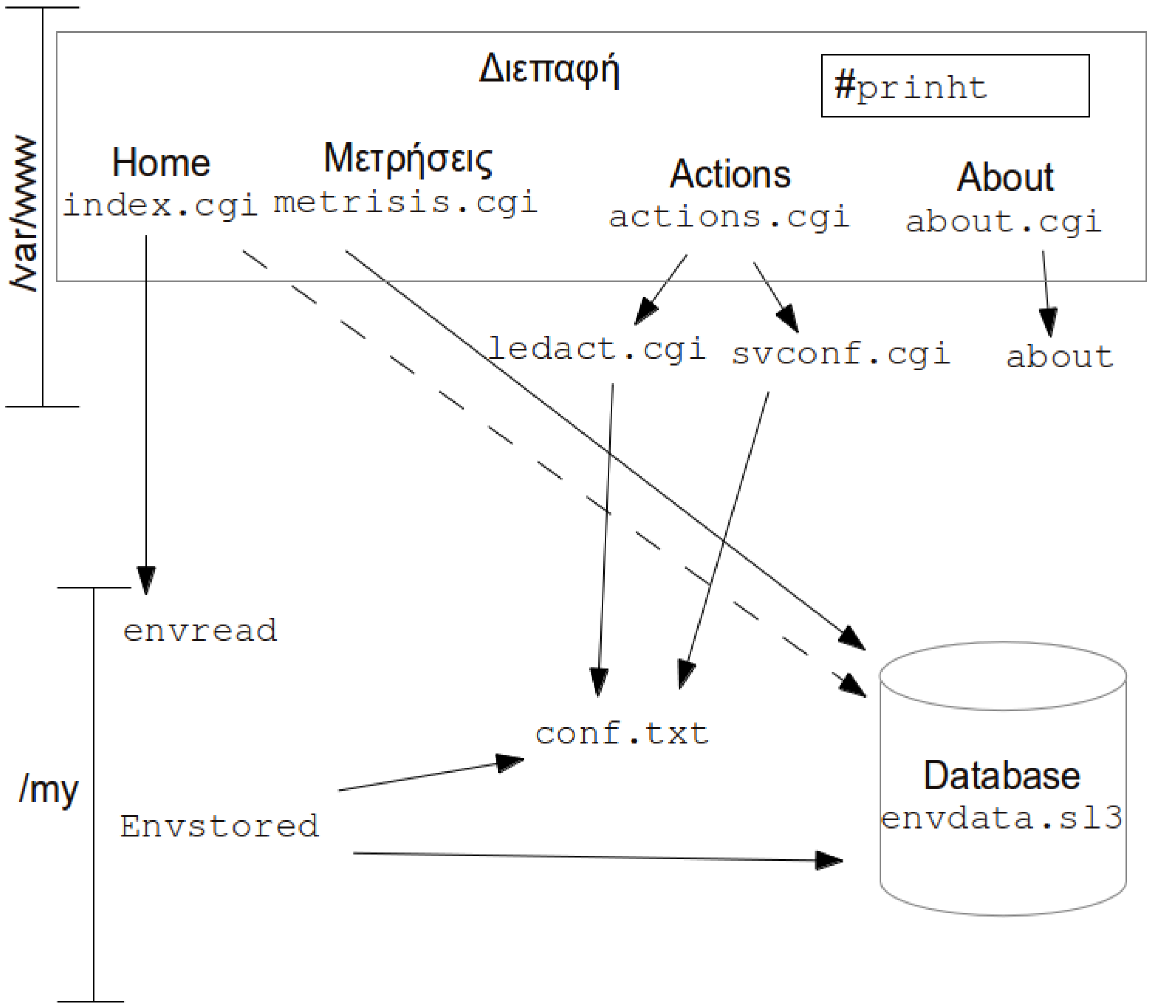
5.1.2 Σχεδιασμός

Το σχήμα (Εικόνα 9: Σχεδιασμός) βοηθάει να κατανοήσουμε πώς συνδέονται τα προγράμματα και τα αρχεία που απαρτίζουν το σύστημα ελέγχου περιβαλλοντικών μεταβλητών. Τα ονόματα αρχείων είναι γραμμένα με σταθερού πλάτους γραμματοσειρά “FreeMono”. Τα βέλη δείχνουν τις αλληλεπιδράσεις αρχείων. Στα αριστερά φαίνεται ο φάκελος στον οποίο περιέχονται τα αρχεία. Οι δύο φάκελοι που χρησιμοποιούνται είναι ο “/my” για τα προγράμματα και αρχεία παρασκηνίου, και ο “/var/www” για τη διεπαφή.

Ξεκινώντας από πάνω, μέσα στο τετράγωνο “Διεπαφή”, βλέπουμε τα τέσσερα αρχεία cgi που απαρτίζουν τις σελίδες της διεπαφής και τις ονομασίες τους στο μενού. Στο εξής θα αναφερόμαστε σε αυτά ως βασικά cgi. Το πρόγραμμα prinht καλείται στην αρχή κάθε ενός από αυτά τα cgi ώστε να τυπώσει το πάνω μέρος του html κώδικα. Αυτό περιλαμβάνει τα μενού, css και javascript.

Έξω από το το τετράγωνο διεπαφή, αλλά μέσα στο φάκελο “/var/www” βρίσκονται τα αρχεία που συμπληρώνουν τη λειτουργία των σελίδων. Τα envread και about παρέχουν περιεχόμενο στη σελίδα, ενώ τα ledact.cgi και svconf.cgi καλούνται για να εκτελέσουν εργασίες και να επιστρέψουν τον έλεγχο στο actions.cgi.

Πιο κάτω φαίνονται τα προγράμματα του φακέλου /my. Εκεί βρίσκονται όσα αρχεία δεν είναι τόσο ορατά από τη διεπαφή. Το envstored είναι το πρόγραμμα που τρέχει στο παρασκήνιο, καταγράφει τις μετρήσεις και ελέγχει τα LED. Το conf.txt είναι το αρχείο ρυθμίσεων που φέρνει σε επαφή το actions.cgi και το envstored. Το envdata.sl3 είναι η βάση δεδομένων. Για την SQLite η κάθε βάση δεδομένων είναι ένα αρχείο.



5.2 Σχεδιασμός αναλυτικά

Εδώ θα εξετάσουμε τη λειτουργία του κάθε προγράμματος αναλυτικά.

5.2.1 Βοηθητικά προγράμματα

Το prinht είναι ένα εκτελέσιμο που τυπώνει το πάνω μέρος του κώδικα τις ιστοσελίδας, μέχρι και την αρχή του <body>. Το κάθε βασικό cgi (5.3.3 Προγραμματισμός CGI) εκτελεί αυτό το πρόγραμμα και τυπώνει την έξοδο του, πριν τυπώσει οποιοδήποτε άλλον κώδικα html. Έτσι το μενού, και οι αρχικές ρυθμίσεις για css και javascript είναι κοινά και για τις τέσσερις σελίδες. Είναι η ίδια τακτική με το header.php που χρησιμοποιείται συχνά σε php σελίδες.

Το conf.txt είναι το αρχείο ρυθμίσεων όπου αποθηκεύονται τα ανώτερα και κατώτερα όρια για τα LED, καθώς και οδηγίες προς το παρασκήνιο για το αν πρέπει να τα χειριστεί αυτόματα ή όχι. Το 0 σημαίνει αυτόματο, το 1 αναμμένο, το -1 σβηστό. Ένα δείγμα του conf.txt ειναι αυτό:

```
max tempetarure     27
max humidity   56
min temperature     23
min humitity   45
LED1      0
LED2      0
```

Τέλος το envdata.sl3 είναι το αρχείο της βάσης δεδομένων που δημιούργησε η SQLite3. Περιέχει μόνο τον πίνακα envstore. Τα πεδία του πίνακα είναι τα εξής:

time timestamp

heat integer
humidity integer
Για να λειτουργήσει το σύστημα, πρέπει η βάση δεδομένων και ο πίνακας να δημιουργηθούν χειροκίνητα.

5.2.2 envstored

Όπως είπαμε παραπάνω το envstored είναι το βασικό πρόγραμμα που λαμβάνει και αποθηκεύει τις μετρήσεις. Τρέχει από προεπιλογή στο παρασκήνιο, αλλά μπορεί να εκτελεστεί κανονικά για λόγους παρακολούθησης ή αποσφαλμάτωσης. Το σχήμα (Εικόνα 14: ανάλυση envstored) βοηθάει να καταλάβουμε τη λειτουργία του. Με βάση το όρισμα που θα δεχτεί, έχει τρεις λειτουργίες:

- Με το όρισμα -h εμφανίζει ένα κείμενο βοήθειας. Στο κείμενο αυτό αναφέρονται και τα ορίσματα.
- Με το όρισμα -s η κανονική έξοδος του προγράμματος τυπώνεται στο τερματικό. Έτσι μπορούμε να δούμε τυχόν μηνύματα σφάλματος.
- Τέλος χωρίς ορίσματα, το πρόγραμμα μεταφέρεται στο παρασκήνιο, αφού τυπώσει κάποιες πληροφορίες που επιβεβαιώνουν τη σωστή του εκκίνηση.

Μετά την ανάγνωση των ορισμάτων, γίνεται κάποια προετοιμασία που περιλαμβάνει τη ρύθμιση του προσαρμογέα SPI (4.11 SPI) ώστε να μπορεί να συνεργαστεί με τον αισθητήρα και αρχικοποίηση των LED. Τα LED κάθε φορά που εκκινούμε την πλακέτα είναι απενεργοποιημένα. Έτσι το envstored ελέγχει αν είναι ενεργοποιημένα και αν όχι τα ενεργοποιεί πριν τα χρησιμοποιήσει. Ακολουθεί η μεταφορά στο παρασκήνιο αν δεν υπάρχουν ορίσματα, η οποία ελευθερώνει το τερματικό.Έπειτα το envstored μπαίνει σε έναν ατέρμονα βρόγχο. Μέσα σε αυτόν γίνεται η ανάγνωση των μετρήσεων από τον αισθητήρα και η αποθήκευση τους στη βάση δεδομένων, αν αυτές διαφέρουν από τις προηγούμενες τιμές. Επιπλέον, με κάθε ανάγνωση των τιμών γίνεται και ανάγνωση του conf.txt, με βάση το οποίο γίνονται οι απαραίτητες ενέργειες στα LED. Στο τέλος του βρόγχου υπάρχει μια εντολή αναμονής, ώστε η διαδικασία να επαναλαμβάνεται κάθε ένα λεπτό.

5.2.3 Σελίδες

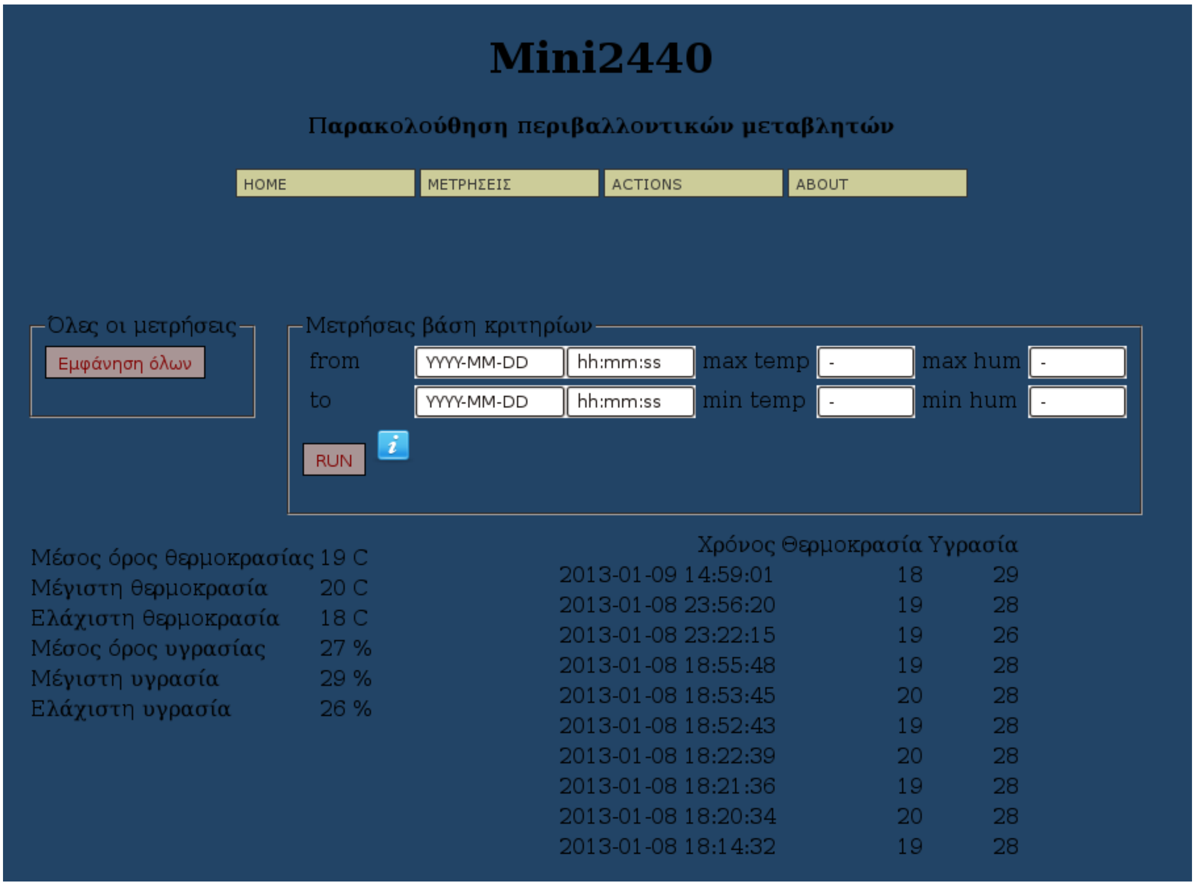
Η σελίδα Home μας πληροφορεί για την τρέχουσα κατάσταση. Συγκεκριμένα μας ενημερώνει για την τρέχουσα ημερομηνία και ώρα (με βάση το ρολόι της πλακέτας) και για την τρέχουσα θερμοκρασία και υγρασία. Επιπλέον, δεδομένης της σημασίας του envstored για την ορθή λειτουργία της διεπαφής, μας ενημερώνει αν το envstored είναι σε λειτουργία ή όχι. Αν το envstored είναι ανενεργό, τρέχουσες περιβαλλοντικές μεταβλητές διαβάζονται μέσω του envread, ειδάλλως εμφανίζεται η τελευταία μέτρηση.



Εικόνα 10: διεπαφή αρχική

Η σελίδα metrisis εμφανίζει τις μετρήσεις που έχουν αποθηκευτεί στη βάση δεδομένων. Στο πάνω μέρος υπάρχουν τα εξής: η φόρμα αναζήτησης με κριτήρια και η επιλογή "όλες οι μετρήσεις". Με τη φόρμα κριτηρίων μπορούμε να περιορίσουμε τα αποτελέσματα της αναζήτησης με βάση ελάχιστη και μέγιστη: ημερομηνία, ώρα, θερμοκρασία, υγρασία. Ουσιαστικά δηλαδή με όλα τα πιθανά κριτήρια. Η επιλογή "όλες οι μετρήσεις" όπως υποδηλώνει το όνομα της εμφανίζει όλες τις καταγραφές που έχουν γίνει.

Πιο κάτω στα αριστερά εμφανίζεται ένας συνοπτικός πίνακας με το μέσο όρο, το μέγιστο και το ελάχιστο, τόσο για τη θερμοκρασία όσο και για την υγρασία των επιλεγμένων αποτελεσμάτων. Στα αριστερά εμφανίζονται τα αποτελέσματα με τη μορφή χρόνος λήψης – θερμοκρασία -υγρασία. Ακριβώς από κάτω αναγράφεται το πλήθος των αποτελεσμάτων και εμφανίζονται κουμπιά "επόμενο", "προηγούμενο" αν και εφόσον κάτι τέτοιο είναι απαραίτητο. Τα αποτελέσματα εμφανίζονται σε σελίδες είκοσι αποτελεσμάτων η κάθε μια. Αν δεν δοθούν κριτήρια, εμφανίζονται οι δέκα τελευταίες μετρήσεις.



Εικόνα 11: διεπαφή μετρήσεις

Στη σελίδα Actions μπορούμε να δούμε την κατάσταση των LED και να την επηρεάσουμε. Στο πάνω μέρος υπάρχουν δύο εικόνες που γράφουν “ON” ή “OFF”, ανάλογα με την κατάσταση του κάθε LED. Ακριβώς κάτω από την κάθε μία βλέπουμε τις επιλογές “on”, “off” και “auto”. Οι επιλογές “on”, “off” παρακάμπτουν τη λειτουργία “auto”. Η επιλεγμένη φαίνεται με πράσινο χρώμα. Πιο κάτω μπορούμε να δούμε και να αλλάξουμε τα μέγιστα και τα ελάχιστα όρια για θερμοκρασία και την υγρασία, ορίζοντας έτσι τη συμπεριφορά του auto.



Εικόνα 12: διεπαφή ενέργειες

Σε κάποιες σελίδες υπάρχει ένα εικονίδιο βοήθειας. Αν το πατήσουμε εμφανίζει ένα πλαίσιο με οδηγίες όπως φαίνεται στην εικόνα (Εικόνα 13: διεπαφή βοήθεια).



Εικόνα 13: διεπαφή βοήθεια

5.3 Κώδικας και αλγόριθμοι

Σε αυτό το υπό-κεφάλαιο θα αναλύσουμε τα πιο σημαντικά κομμάτια κώδικα που χρησιμοποιήθηκαν. Όπως έχει αναφερθεί σε προηγούμενο κεφάλαιο, τόσο η διεπαφή όσο και το πρόγραμμα envstored έχουν γραφεί στη C. Έτσι έχουμε μια ομοιομορφία στον κώδικα. Για την παραγωγή των εκτελέσιμων χρησιμοποιήθηκε ο cross-compiler που βρίσκεται στη σελίδα της πλακέτας (arm-2008q3). Για τη χρήση SQLite3 χρειάστηκε να προσθέσουμε κάποιες βιβλιοθήκες, οι οποίες αναφέρονται στην περιγραφή της SQLite3 (4.10 SQLite3).

5.3.1 Έτοιμος κώδικας

Όπως είναι φυσικό, κάποια κομμάτια κώδικα, είναι εύκολο να βρεθούν στο διαδίκτυο και να χρησιμοποιηθούν με λίγη ή καθόλου παραμετροποίηση. Η τακτική αυτή έχει σίγουρα αποτελέσματα, αφού ο κώδικας που θα χρησιμοποιήσουμε έχει δοκιμαστεί και από άλλους χρήστες προκειμένου να τελειοποιηθεί. Εδώ θα αναλύσουμε αυτά τα έτοιμα κομμάτια που χρησιμοποιήθηκαν.

Για τη λήψη δεδομένων που στάλθηκαν μέσω της GET στα cgi χρησιμοποιήθηκε έτοιμος κώδικας, χωρίς αλλαγές. Ο κώδικας αυτός περιλαμβάνει της συναρτήσεις char x2c(char *what), void unescape_url(char *url) και char **getcgivars(). Για να λειτουργήσει απαιτεί τις βιβλιοθήκες stdio.h, string.h και stdlib.h. Μετατρέπει τη συμβολοσειρά εισόδου σε ζεύγη του τύπου όνομα πεδίου - τιμή. Ο κώδικας βρέθηκε στη σελίδα “http://www.jmarshall.com/easy/cgi/”. Για να τον χρησιμοποιήσουμε καλούμε τη συνάρτηση getcgivars(). Έπειτα χρησιμοποιούμε τον πίνακα χαρακτήρων δύο διαστάσεων getcgivars, όπου getcgivars[i] είναι το όνομα του πεδίου και getcgivars[i+1] η τιμή του. Παράρτημα (Ανάγνωση ορισμάτων get:)

Επιπλέον το αρχείο spidev_test.c, που βρίσκεται στο φάκελο “Documentation/spi” του kernel, χρησιμοποιήθηκε σα βάση για την ανάπτυξη του envstored. Η συνάρτηση transfer υπέστη αλλαγές που περιγράφονται (5.3.2 Επικοινωνία με τον αισθητήρα) και χρησιμοποιείται για τη λήψη

μετρήσεων, τόσο από το envstored, όσο και από το envread. Επίσης η συνάρτηση λήψης ορισμάτων που χρησιμοποιείται στο envstored, με μόνη αλλαγή τα ορίσματα που δέχεται.

5.3.2 Επικοινωνία με τον αισθητήρα

Η επικοινωνία με τον αισθητήρα επιτυγχάνεται με τη χρήση του πρωτοκόλλου SPI (4.11 SPI). Προκειμένου να χρησιμοποιήσουμε το SPI μέσο της C, υπάρχουν δύο τρόποι. Ο πρώτος είναι η βιβλιοθήκη “spi.h”. Με αυτό τον τρόπο έχουμε επικοινωνία με τη συσκευή SPI σε χαμηλό επίπεδο. Μπορεί να χρησιμοποιηθεί ακόμα και για τη συγγραφή οδηγών SPI (drivers). Απ' τη στιγμή όμως που ο πυρήνας μας έχει οδηγούς για την SPI, είναι προτιμότερο να χρησιμοποιήσουμε η βιβλιοθήκη “spidev.h”. Αυτός είναι ένας υψηλότερου επιπέδου τρόπος, που μας απαλλάσσει εν μέρει από την πολυπλοκότητα του υλικού.

Ως βάση για τον προγραμματισμό της επικοινωνίας με τον αισθητήρα, χρησιμοποιήθηκε το “spidev_test.c” από το φάκελο “Documentation/spi” του πυρήνα. Το πρόγραμμα αυτό χρησιμοποιεί την “spidev.h”, για να στείλει έναν πίνακα δεδομένων μέσω της spi και να παραλάβει έναν σαν απάντηση. Ο πίνακας αυτός είναι προκαθορισμένος. Η μόνη του χρήση είναι να ελέγξουμε την ορθή λειτουργία της συσκευής. Η συσκευή, η ταχύτητα, και άλλοι παράμετροι, μπορούν να καθοριστούν με ορίσματα. Αν βραχυκυκλώσουμε τον ακροδέκτη αποστολής δεδομένων με αυτόν της λήψης, και πάρουμε σαν είσοδο τα δεδομένα του πίνακα, σημαίνει πως το πρόγραμμα δουλεύει σωστά.

Το spidev_test ερμηνεύει τα ορίσματα χάρη στη συνάρτηση parse_opts. Με βάση αυτά γεμίζει τις μεταβλητές device, speed, delay, bits και mode. Η μεταβλητή mode είναι ένας ακέραιος 8 bit στον οποίο αποθηκεύεται ό,τι έχει σχέση με το SPI mode. Τα δύο λιγότερο σημαντικά bit καθορίζουν το mode με τη σύντομη έννοια, ενώ τα υπόλοιπα ορίζουν αν η ενεργοποίηση του σκλάβου θα γίνεται στην υψηλή ή τη χαμηλή τάση, αν προηγείται το πιο σημαντικό ή το λιγότερο σημαντικό bit και άλλες επιλογές που έχουν να κάνουν με τη συνεργασία αφέντη σκλάβου.

Στην main ανοίγουμε το αρχείο που ελέγχει τη συσκευή ("/dev/spidev0.0") και με τη βοήθεια της εντολής ioctl ρυθμίζουμε το mode, το μέγεθος λέξης (bits per word) και την ταχύτητα. Η εντολή ioctl είναι χρήσιμη σε προγραμματισμό χαμηλού επιπέδου. Η πρώτη της παράμετρος είναι, περιγραφέας αρχείου (open file descriptor), η δεύτερη περιγράφει τη δράση της εντολής ,ενώ μπορεί να χρειαστούν επιπλέον ορίσματα.

Η συνάρτηση transfer αναλαμβάνει τη μεταφορά δεδομένων. Από προεπιλογή το μέγεθος λέξεις είναι 8 bit. Οπότε ως μήνυμα προς αποστολή ορίζει έναν πίνακα ακεραίων μεγέθους ενός byte ο καθένας, και τον γεμίζει με αριθμούς στο δεκαεξαδικό. Έπειτα δημιουργεί τη δομή spi_ioc_transfer και ορίζει μια μεταβλητή tr αυτού του τύπου. Η δομή αυτή είναι κατασκευασμένη με βάση τις απαιτήσεις της ioctl. Περιέχει πληροφορίες για την επικοινωνία αφέντη σκλάβου, όπως η ταχύτητα και το μέγεθος λέξης, καθώς και τους πίνακες αποστολής και λήψης δεδομένων (tx και rx αντίστοιχα). Έτσι, μετά τη δημιουργία της tr καλείται η ioctl με ορίσματα τη συσκευή SPI, SPI_IOC_MESSAGE(1) και &tr. Τα ληφθέντα δεδομένα τα διαβάζουμε από τον πίνακα rx με μία απλή εντολή επανάληψης (for).

Για τις ανάγκες της εργασίας, η δυνατότητα αλλαγής της συσκευής και του SPI mode είναι περιττές. Έτσι, αντί να εισάγονται με ορίσματα, καθορίζονται στατικά στις δηλώσεις των μεταβλητών. Επιπλέον για τον αισθητήρα DC-SS500 που χρησιμοποιήθηκε, τόσο η ερώτηση όσο και η απάντηση είναι ένα μόνο byte. Γιαυτό χρησιμοποιούμε μόνο το πρώτο κελί του πίνακα αποστολής και λήψης. Η εντολή επανάληψης κατά την ανάγνωση λοιπόν αφαιρέθηκε. Επιπλέον η μεταβλητή delay άλλαξε από uint16_t (ακέραιος 16 bit χωρίς πρόσημο) σε uint32_t (ακέραιος 32 bit χωρίς πρόσημο) για να καλυφθούν οι απαιτήσεις του αισθητήρα.

Ένα πρόβλημα που παραμένει είναι, ότι για κάθε ερώτηση που υποβάλουμε στον αισθητήρα, λαμβάνουμε την απάντηση της προηγούμενης ερώτησης. Συνεπώς η πρώτη λέξη που θα πάρουμε μετά την εκκίνηση είναι άχρηστη. Για την επίλυση του προβλήματος αποστέλλεται μια επιπλέον ερώτηση στον αισθητήρα, ώστε να λαμβάνουμε πάντα την απάντηση της προηγούμενης ερώτησης.

Συγκεκριμένα, πρέπει να στείλουμε 0xA0 (δεκαεξαδικό 160) για να πάρουμε την υγρασία και 0xB0 (δεκαεξαδικό 176) για να πάρουμε τη θερμοκρασία. Εκτελούμε λοιπόν την εντολή ioctl τρεις φορές μέσα σε μία εντολή επανάληψης. Την πρώτη φορά το tx έχει τιμή 0xA0 και κάθε φορά μετά

την ανάγνωση των δεδομένων προστίθεται στο tx ο αριθμός 16. Έτσι στέλνονται διαδοχικά 0xA0, 0xB0 και 0xC0. Κατά τη δεύτερη εκτέλεση διαβάζουμε την υγρασία και κατά την τρίτη την υγρασία, αγνοώντας την απάντηση της τρίτης εντολής. Η εντολή 0xC0 είναι υπαρκτή και ρωτάει τον αισθητήρα αν είναι απασχολημένος. Αν απαντήσει 0xAF σημαίνει ότι δεν είναι. Γιαυτό αν τρέξουμε το envstored με την επιλογή -s, ώστε να εμφανίζει πληροφορίες στην οθόνη, η πρώτη απάντηση σε κάθε κύκλο επαναλήψεων είναι 175 (δεκαδικός του 0xAF).

Οι λεπτομέρειες που αφορούν τις μεταβλητές delay, speed και mode, αναλύονται στο κεφάλαιο (4.12 SPI στην περίπτωση μας).

5.3.3 Προγραμματισμός CGI

Ένας τρόπος παραγωγής δυναμικών ιστοσελίδων είναι τα αρχεία CGI. Μπορούν να είναι γραμμένα σχεδόν σε οποιαδήποτε γλώσσα προγραμματισμού ή γλώσσα σεναρίων. Συχνά αναφερόμαστε σε αυτά ως σενάρια cgi (cgi scripts), ενώ μπορούν κάλλιστα να είναι προγράμματα. Αυτό συμβαίνει γιατί συνήθως γράφονται σε PERL που είναι γλώσσα σεναρίων. Στην περίπτωση μας βέβαια θα γραφούν στη C. Σχεδόν όλοι οι web servers τα υποστηρίζουν.

Προκειμένου να εκτελέσουμε μια ιστοσελίδα σε cgi, πρέπει αρχικά να δούμε τις ρυθμίσεις του server μας και πιθανόν να τις αλλάξουμε. Οι περισσότεροι servers εκτελούν cgi μόνο από ένα συγκεκριμένο φάκελο, ο οποίος ίσως δε βρίσκεται σε βολική θέση. Αφού βρούμε ή ορίσουμε το φάκελο για τα cgi (συνήθως λέγεται cgi-bin) μπορούμε να τοποθετήσουμε μέσα σε αυτόν τα cgi αρχεία. Από εκεί μπορούμε να τα καλέσουμε σαν ιστοσελίδες. Η διεύθυνση θα έχει μορφή του τύπου <διεύθυνση server>/cgi/<όνομα αρχείου cgi> (ανάμεσα στη διεύθυνση και το όνομα αρχείου αντί για cgi μπορεί να χρειάζεται cgi-bin ή κάτι άλλο ανάλογα με τις ρυθμίσεις).

Ο cherokee από προεπιλογή αναζητούσε τα αρχεία cgi στο φάκελο “/usr/lib/cgi-bin/” και για να τα καλέσουμε μετά τη διεύθυνση της πλακέτας έπρεπε να προσθέσουμε “/cgi-bin/αρχείο.cgi”. Για λόγους ευκολίας όρισα ως φάκελο για cgi τον “/var/www” που χρησιμοποιείται ούτως ή άλλως για την αποθήκευση ιστοσελίδων. Έτσι θα μπορούσε να συνδυαστεί ευκολότερα η χρήση cgi με HTML αρχείων. Επιπλέον άλλαξα το “/cgi-bin/αρχείο.cgi” σε “/cgi/αρχείο.cgi”.

Σε γενικές γραμμές, ο τρόπος χρήσης αυτών των αρχείων, θυμίζει έντονα τη λογική της php. Κάθε αρχείο cgi μπορεί να δεχτεί ορίσματα μέσω των get και post. Επεξεργάζεται τα δεδομένα με βάση τις δυνατότητες της γλώσσας στην οποία γράφηκε. Τέλος, τυπώνει των κώδικα της σελίδας που θα εμφανίσει ο server. Σε αντίθεση με την php δε μπορούμε να παρεμβάλλουμε κώδικα HTML αναμεμιγμένο με τον κώδικα του προγράμματος. Αν και με τη χρήση SSI(Server Side Includes) μπορούμε να ενσωματώσουμε την κλήση ενός cgi στον HTML κώδικα.

Για τις ανάγκες της εργασίας χρειάστηκε να τυπωθούν σε κάποιες περιπτώσεις σχετικά μεγάλα στατικά κομμάτια κώδικα HTML. Υπάρχουν δύο τρόποι να γίνει αυτό. Ο ένας είναι να τυπώσουμε τον κώδικα με χρήση printf, puts. Αυτός ο τρόπος χρησιμοποιείται στο αρχείο prinht το οποίο καλείται στην αρχή κάθε cgi και τυπώνει το πάνω μέρος του κώδικα (head, menu). Ο δεύτερος τρόπος είναι να δημιουργήσουμε ένα αρχείο κειμένου, που θα περιέχει τον κώδικα της σελίδας. Έπειτα καλούμε αυτό το κομμάτι και το τυπώνουμε στο σημείο που βολεύει. Έτσι εμφανίζεται στο σώμα της σελίδας about.

Γράφοντας ένα cgi, το πρώτο που πρέπει να τυπώσουμε είναι “Content-type: text/html” ακολουθούμενο από μία κενή γραμμή. Στη C αυτό το κάνουμε με τη γραμμή 'printf("Content-Type: text/html;charset=utf-8\n\n");'. Το charset=utf-8 δηλώνει ότι η κωδικοποίηση των χαρακτήρων είναι utf-8. Για τη λήψη των ορισμάτων μπορούμε να χρησιμοποιήσουμε έτοιμες συναρτήσεις (5.3.1 Έτοιμος κώδικας).

5.3.4 Αλλαγή αρχείων κειμένου

Σε κάποιες περιπτώσεις όπως η αποθήκευση ρυθμίσεων και η αλλαγή κατάστασης των LED, χρειάστηκε να αλλάξουμε το περιεχόμενο αρχείων κειμένου μέσω της C. Για μικρού όγκου αλλαγές, όπως ο χειρισμός τον LED όπου το περιεχόμενο του αρχείου είναι μόνο ένας αριθμός, χρησιμοποιήθηκε η εντολή system. Αυτή η εντολή μας δίνει τη δυνατότητα να τρέξουμε εντολές τερματικού μέσα από τη C. Έτσι η γραμμή “system("echo 0 > /sys/class/gpio/gpio38/value");” ανάβει

το LED 2. Ο τρόπος αυτός, απαιτεί το πρόγραμμα να τρέχει σε Linux, είναι όμως απλός και άμεσος . Βέβαια αν και είναι κατάλληλος για πολύ μικρά αρχεία, γίνεται προβληματικός με έστω και λίγο μεγαλύτερα.

Αντιθέτως για αρχεία με περισσότερους χαρακτήρες και ειδικά αν υπάρχει αλλαγή γραμμής, όπως το αρχείο conf.txt στο οποίο αποθηκεύονται οι ρυθμίσεις που αφορούν τα όρια και τη λειτουργία των LED αξιοποιούμε τον παραδοσιακό τρόπο που προσφέρει η C. Ανοίγουμε το αρχείο με ένα δείκτη σε αυτό, το φορτώνουμε σε ένα πίνακα χαρακτήρων δύο διαστάσεων, και το επεξεργαζόμαστε.

5.3.5 Μεταφορά στο παρασκήνιο

Τα προγράμματα που εκτελούνται στο παρασκήνιο, ειδικά στο Linux, ονομάζονται δαίμονες (daemons), επομένως η διαδικασία μεταφοράς ενός προγράμματος στο παρασκήνιο λέγεται demonize. Το όνομα αυτό μάλλον τους δόθηκε χαριτολογώντας, καθώς η εκτέλεση τους είναι διακριτική και το μόνο που βλέπει ο απλός χρήστης είναι το αποτέλεσμα της εργασίας τους. Σε άλλα λειτουργικά, πιθανό για να μην προκαλούν, ονομάζονται υπηρεσίες (services), έργα (tasks) ή εργασίες φαντάσματα (ghost jobs)

Η διαδικασία μεταφοράς ενός προγράμματος στο παρασκήνιο σε περιβάλλον Linux περιγράφεται εδώ (http://www-theorie.physik.unizh.ch/~dpotter/howto/daemonize?do=show). Μάλιστα παρουσιάζονται δύο λύσεις, μια πιο στοιχειώδης και μια πιο πλήρης. Για τους σκοπούς της εργασίας χρησιμοποιήθηκε η απλή λύση με κάποια στοιχεία από την πιο πολύπλοκη.

Τα βήματα που χρησιμοποιήθηκαν για μεταφορά του προγράμματος στη C στο παρασκήνιο είναι:

1. Διακλάδωση (fork) της εφαρμογής, με τη διαδικασία αυτή, δημιουργούμε μια δεύτερη διεργασία, όμοια με την πρώτη, έτοιμη να συνεχίσει την εκτέλεση της από το ίδιο σημείο. Έπειτα η γονική διεργασία τερματίζεται.
2. Η νέα διεργασία αποδεσμεύεται από την παλιά με τη χρήση της εντολής setsid.
3. Ορίζουμε τα δικαιώματα αρχείων και φακέλων που πιθανό να δημιουργηθούν από τη διεργασία με την εντολή umask.
4. Αλλάζουμε τον κατάλογο εκτέλεσης του προγράμματος σε κάποιον άλλο, συνήθως στον root (/), έτσι ο κατάλογος βρίσκετε το εκτελέσιμο αποδεσμεύεται.
5. Ξανά ανοίγουμε της κανονικές εισόδους και εξόδους του αρχείου (stdin, stdout και stderr), αναδρομολογώντας τες στο "/dev/null". Με αυτό τον τρόπο η διεργασία δε δέχεται είσοδο και η έξοδος της δεν εμφανίζεται πουθενά.
6. Ελέγχουμε αν ο δαίμονας τρέχει ήδη με την εντολή getppid.

Τα βήματα 1-5 είναι αναγκαία, ενώ το έκτο προαιρετικό.

5.4 Απαιτήσεις συστήματος

Η χρήση της εφαρμογής δεν περιορίζεται την πλακέτα mini2440, αλλά σε οποιαδήποτε συσκευή η οποία, πληρεί ορισμένα κριτήρια. Αρχικά το λειτουργικό σύστημα που θα φιλοξενήσει την εφαρμογή πρέπει να είναι Linux. Με λίγες αλλαγές ίσως, μπορεί να δουλέψει και σε άλλα λειτουργικά βασισμένα στο Unix. Πρέπει να υπάρχουν εγκατεστημένα: ένας διακομιστής ιστού (web server), που να υποστηρίζει αρχεία cgi και η βάση δεδομένων SQLite3. Επιπλέον ο πυρήνας πρέπει να είναι εξοπλισμένος με τους οδηγούς (drivers) για GPIO και SPI. Τέλος, απαραίτητος είναι ένας αισθητήρας θερμοκρασίας και υγρασίας DC-SS500 της Sure Electronics και τα κατάλληλα καλώδια σύνδεσης. Αν χρησιμοποιηθεί άλλος αισθητήρας θα πρέπει να υποστηρίζει πρωτόκολλο SPI και να γίνουν οι απαραίτητες αλλαγές στα προγράμματα envstored και envread, ώστε να συνεργάζονται.

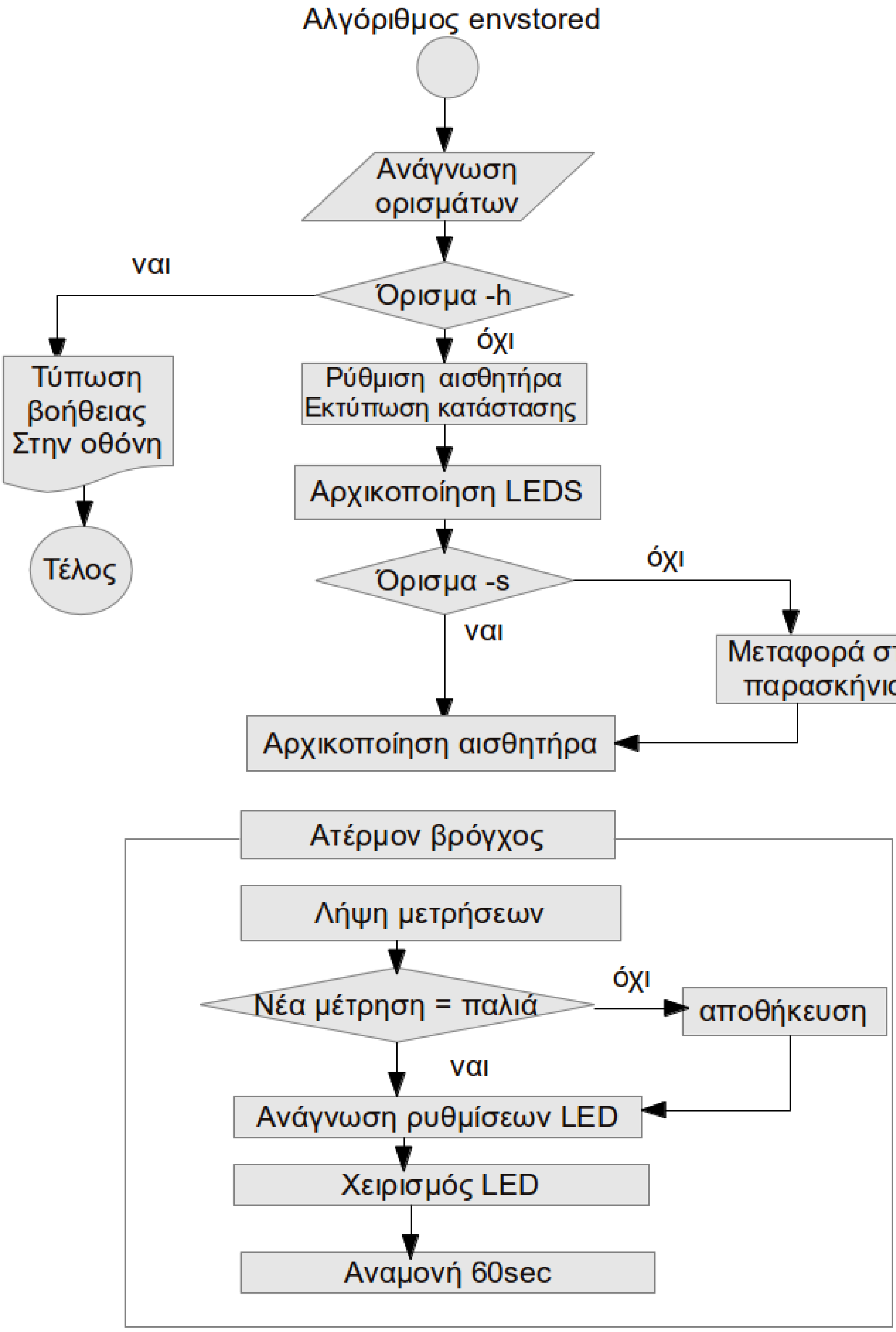
Για τη mini2440 δε βρήκα κάποιο καλώδιο που να ταιριάζει άμεσα στον αισθητήρα. Έτσι με τη βοήθεια ενός φίλου που έχει τα εργαλεία, φτιάξαμε ένα. Τα LED που χρησιμοποίησα είναι ενσωματωμένα στην πλακέτα. Για τη σύνδεση άλλων συσκευών (6.1 Αποτελέσματα) περισσότερα καλώδια θα ήταν απαραίτητα.

5.5 Ανάλυση envstored

- Ανάγνωση ορισμάτων με τη χρήση της συνάρτησης parse_opts. Όπως αναφέρθηκε στο κεφάλαιο έτοιμος κώδικας (5.3.1 Έτοιμος κώδικας), η συνάρτηση αυτή υπήρχε στο

πρόγραμμα spidev_test.c που χρησιμοποιήθηκε σαν βάση για το envstored. Απλώς αλλάχθηκαν τα ορίσματα και οι ενέργειές τους. Τα επιτρεπτά ορίσματα είναι (-s –scrn) και (-h –help). Το πρώτο κάνει το πρόγραμμα να μη μεταφερθεί στο παρασκήνιο, ώστε να δούμε την έξοδο. Αυτό είναι χρήσιμο για έλεγχο ορθής λειτουργίας και εντοπισμό λαθών. Το δεύτερο εμφανίζει ένα σύντομο κείμενο βοήθειας.

- Αν το όρισμα είναι -h ή –help, το κείμενο βοήθειας εμφανίζεται και τερματίζεται το πρόγραμμα.
- Διαφορετικά εκτελείται το επόμενο βήμα που είναι η ρύθμιση του πρωτοκόλλου SPI ώστε να είναι συμβατό με τον αισθητήρα. Αυτό γίνεται ανοίγοντας τον οδηγό του αισθητήρα σαν αρχείο "fd = open(device, O_RDWR);" και χρησιμοποιώντας το δείκτη fd σε έξι διαδοχικές ioctl. Αυτές ορίζουν την ταχύτητα, το μέγεθος λέξης και το mode, χωριστά για την είσοδο και για την έξοδο δεδομένων. Τυπώνονται πληροφορίες σχετικά με τις μεταβλητές που χρησιμοποιήθηκαν.
- Ενεργοποιούνται τα LED αν δεν είναι ήδη ενεργά. Τυπώνονται πληροφορίες.
- Εάν δεν έχει δοθεί όρισμα -s ή –scrn, το πρόγραμμα μεταφέρεται στο παρασκήνιο (5.3.5 Μεταφορά στο παρασκήνιο).
- Γίνεται αρχικοποίηση του αισθητήρα (συνάρτηση initsensor). Η πρώτη μέτρηση που λαμβάνουμε από τον αισθητήρα κάθε φορά που ενεργοποιείται η πλακέτα, είναι λανθασμένη. Για να προσπεράσουμε το πρόβλημα, κάθε φορά που εκκινείται το envstored, πραγματοποιούμε τουλάχιστο μία μεταφορά δεδομένων χωρίς να λάβουμε υπόψιν τα αποτελέσματα.
- Εισαγωγή σε ατέρμονα βρόγχο. Εφόσον θέλουμε το πρόγραμμα να εκτελείται ασταμάτητα, η συνάρτηση transfer που διαβάζει τα δεδομένα βρίσκεται μέσα σε μία "while(1)". Για να διαβάζονται δεδομένα κάθε ένα λεπτό, μετά την transfer, μέσα στο βρόγχο, υπάρχει η εντολή "sleep(60);". Έτσι το πρόγραμμα περιμένει εξήντα δευτερόλεπτα πριν εκτελέσει ξανά την transfer.
- Εκτελείται η συνάρτηση transfer. Μέσα σε αυτή τη συνάρτηση υπάρχει ο κώδικας επικοινωνίας με τον αισθητήρα (5.3.2 Επικοινωνία με τον αισθητήρα), χάρη στον οποίο παίρνουμε τις μετρήσεις.
- Κάθε νέα μέτρηση, συγκρίνεται με την τελευταία που αποθηκεύτηκε. Αν είναι διαφορετική καλείται η συνάρτηση store που την αποθηκεύει στη βάση δεδομένων (5.2.1 Βοηθητικά προγράμματα). Επιπλέον ενημερώνονται οι μεταβλητές που κρατάνε την τελευταία αποθηκευμένη μέτρηση.
- Διαβάζεται το αρχείο conf.txt (5.2.1 Βοηθητικά προγράμματα) και οι ρυθμίσεις του φορτώνονται σε μεταβλητές.
- Η κατάσταση των LED ενημερώνεται με βάση την τελευταία μέτρηση και τις ρυθμίσεις του conf.txt.
- Αν για κάποιο λόγο βγει από τον βρόγχο, γίνεται ελευθέρωση πόρων και έξοδος.

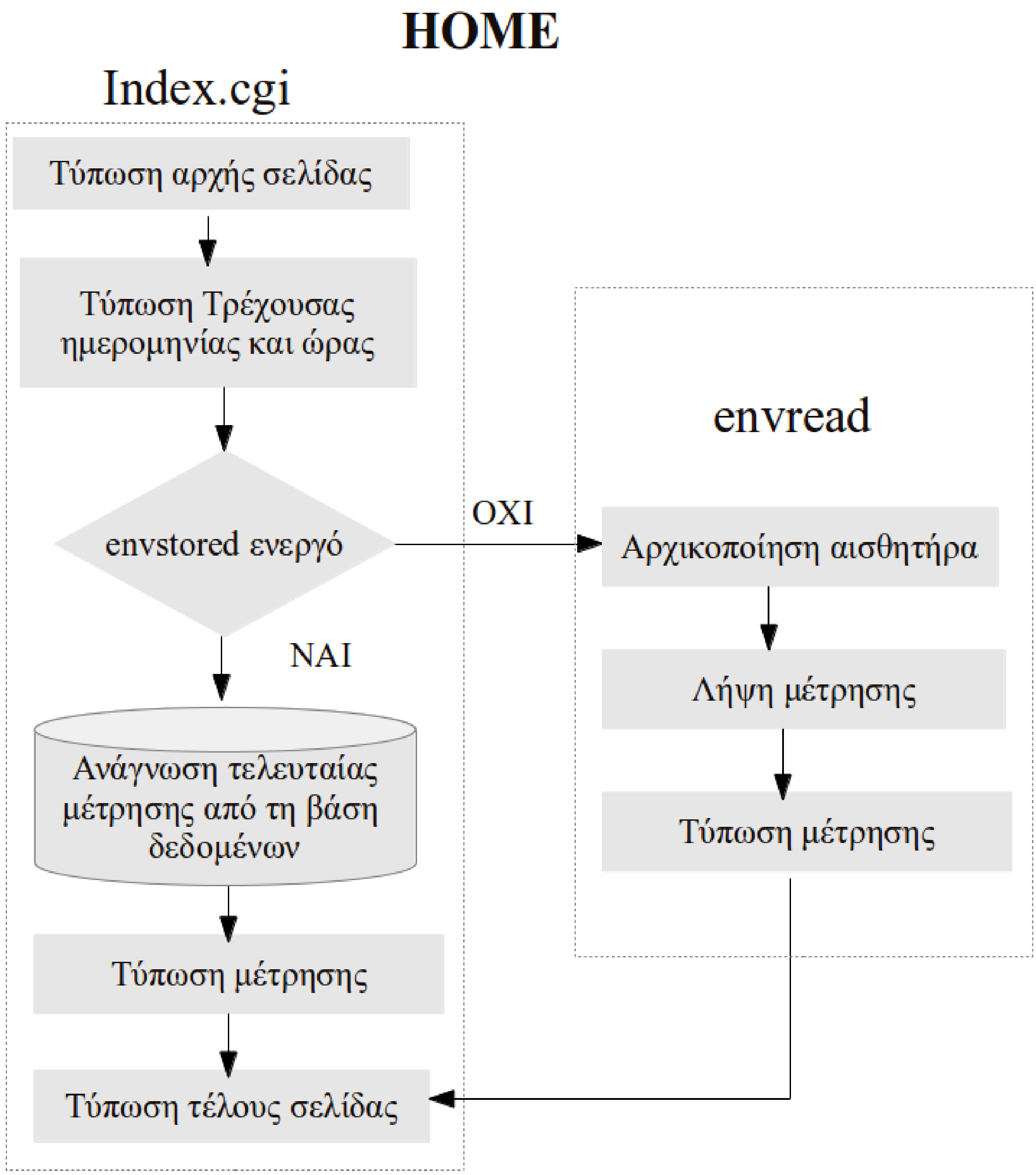


5.6Διεπαφή

Κάποια πράγματα όπως η λειτουργία του prinht έχουν ήδη εξηγηθεί, οπότε θα αναφερθούν πολύ συνοπτικά.

5.6.1 Home “index.cgi”

- τύπωση αρχής σελίδας. Εκτελείται το αρχείο prinht (5.2.1 Βοηθητικά προγράμματα) και τυπώνεται η έξοδος του. Πριν από αυτό έχει τυπωθεί η γραμμή που σηματοδοτεί την έναρξη της εξόδου cgi (5.3.3 Προγραμματισμός CGI).
- Τύπωση τρέχουσας ημερομηνίας και ώρας. Η πληροφορία διαβάζεται με popoen από την εντολή date και τυπώνεται η έξοδος.
- Τύπωση τρέχουσας θερμοκρασίας και υγρασίας. Για να γίνει αυτό, πραγματοποιείται ένας έλεγχος λειτουργίας του envstored, τρέχοντας την εντολή “ps -e | grep envstored” με popen. Αν το envstored είναι ενεργό, διαβάζεται και τυπώνεται η τελευταία μέτρηση από τη βάση δεδομένων. Διαφορετικά εκτελείται το πρόγραμμα envread που παίρνει μετρήσεις από τον αισθητήρα και τυπώνεται η έξοδος του. Το envread δεν αναφέρεται πιο αναλυτικά γιατί, από πλευράς κώδικα, αποτελεί υποσύνολο του envstored.
- Τέλος τυπώνεται αν το envstored είναι σε λειτουργία ή όχι.

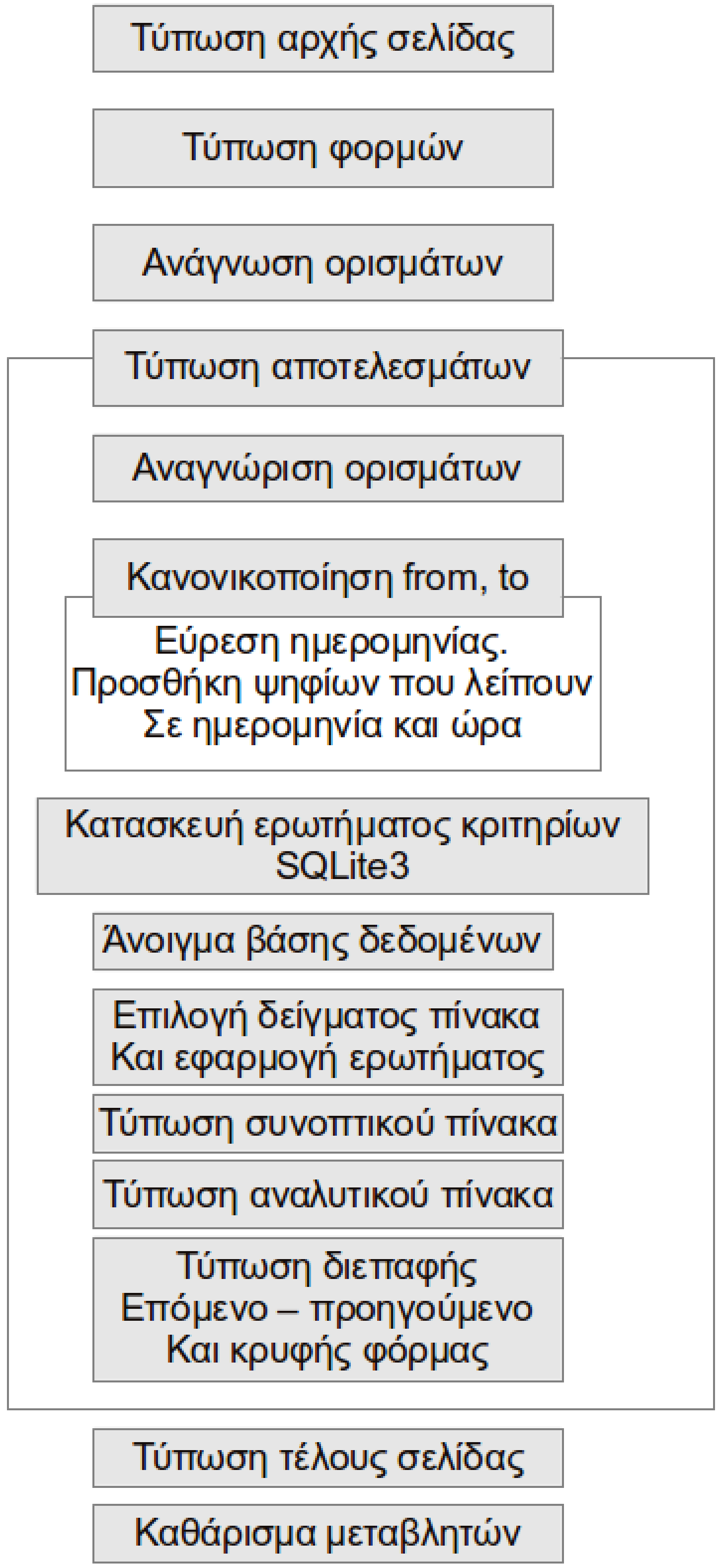


5.6.2 Μετρήσεις metrisis.cgi

- τύπωση αρχής σελίδας.
- Τύπωση φορμών. Τυπώνονται οι δύο φόρμες που καθορίζουν τα κριτήρια αναζήτησης. Η μία δεν παίρνει ορίσματα, αλλά τυπώνει όλες τις μετρήσεις που έχουν ληφθεί. Η δεύτερη, δέχεται τα ορίσματα: ελάχιστη ημερομηνία, ελάχιστη ώρα, ελάχιστη θερμοκρασία, ελάχιστη υγρασία, και τα μέγιστα αυτών. Μπορούμε να συμπληρώσουμε όσα από αυτά τα πεδία θέλουμε προκειμένου να προσαρμόσουμε τα επιθυμητά αποτελέσματα.
- Ανάγνωση ορισμάτων. Αυτό το αρχείο, προκειμένου να εμφανίζει τα αποτελέσματα που επιθυμούμε κάθε φορά, αποστέλλει τα κριτήρια στον εαυτό του.

- Τύπωση αποτελεσμάτων. Με τα βήματα που ακολουθούν, γίνεται η ανάλυση των ορισμάτων και η τύπωση των κατάλληλων αποτελεσμάτων.
- Αναγνώριση ορισμάτων. Αναγνωρίζονται τα ορίσματα που χρησιμοποιήθηκαν με βάση το αν οι τιμές τους είναι διαφορετικές από τις προεπιλεγμένες. Τα προς χρήση ορίσματα φορτώνονται στις κατάλληλες μεταβλητές.
- Κανονικοποίηση from, to. Η ελάχιστη και μέγιστη ημερομηνία και ώρα (αν έχουν συμπληρωθεί) στέλνονται στη συνάρτηση formarg. Εκεί μετράται το πλήθος των ψηφίων της κάθε μεταβλητής και συμπληρώνονται τα ψηφία που λείπουν. Οι περιπτώσεις που καλύπτει, είναι ημερομηνία χωρίς χρονιά (η χρονιά που συμπληρώνεται είναι η τρέχουσα) και ώρα χωρίς δευτερόλεπτα. Σημαντικό είναι η ημερομηνία να χωρίζεται με "-" και η ώρα με ":".
- Κατασκευή ερωτήματος SQLite3. Κατασκευάζεται το μέρος του ερωτήματος που καθορίζει το δείγμα. Αυτό γίνεται προσθέτοντας σε μια συμβολοσειρά την κάθε συνθήκη sql, χωρίζοντάς τες με 'and'.
- Άνοιγμα βάσης δεδομένων (sqlite3_open).
- Με βάση τα ορίσματα, εφαρμόζονται μια σειρά από ερωτήματα στη βάση δεδομένων. Είτε σε όλο τον πίνακα (αν επιλέξαμε εμφάνιση όλων) είτε στο δείγμα που ορίσαμε είτε (χωρίς ορίσματα) στα δέκα τελευταία αποτελέσματα. Τα ερωτήματα αυτά, επιστρέφουν: τις μετρήσεις που έγιναν, τον μέσο όρο, τη μέγιστη και την ελάχιστη θερμοκρασία και υγρασία.
- Τυπώνεται ο συνοπτικός πίνακας με: μέσο όρο, το μέγιστο και το ελάχιστο για θερμοκρασία και υγρασία
- Τυπώνεται ο αναλυτικός πίνακας, με όλες τις μετρήσεις που καλύπτουν τα κριτήρια του δείγματος. Ο πίνακας έχει τη μορφή "χρόνος μέτρησης – θερμοκρασία – υγρασία".
- Σε περίπτωση που ο αναλυτικός πίνακας έχει περισσότερα από είκοσι αποτελέσματα, χωρίζεται σε σελίδες των 20. Μια κρυφή φόρμα τυπώνεται για να διατηρεί τα ορίσματα όταν πατάμε τα κουμπιά επόμενο προηγούμενο.
- Καθάρισμα μεταβλητών. Άδειασμα δυναμικά καταχωρημένων πινάκων και στοιχείων της SQLite.
- Τύπωση τέλους σελίδας.

metrisis.cgi

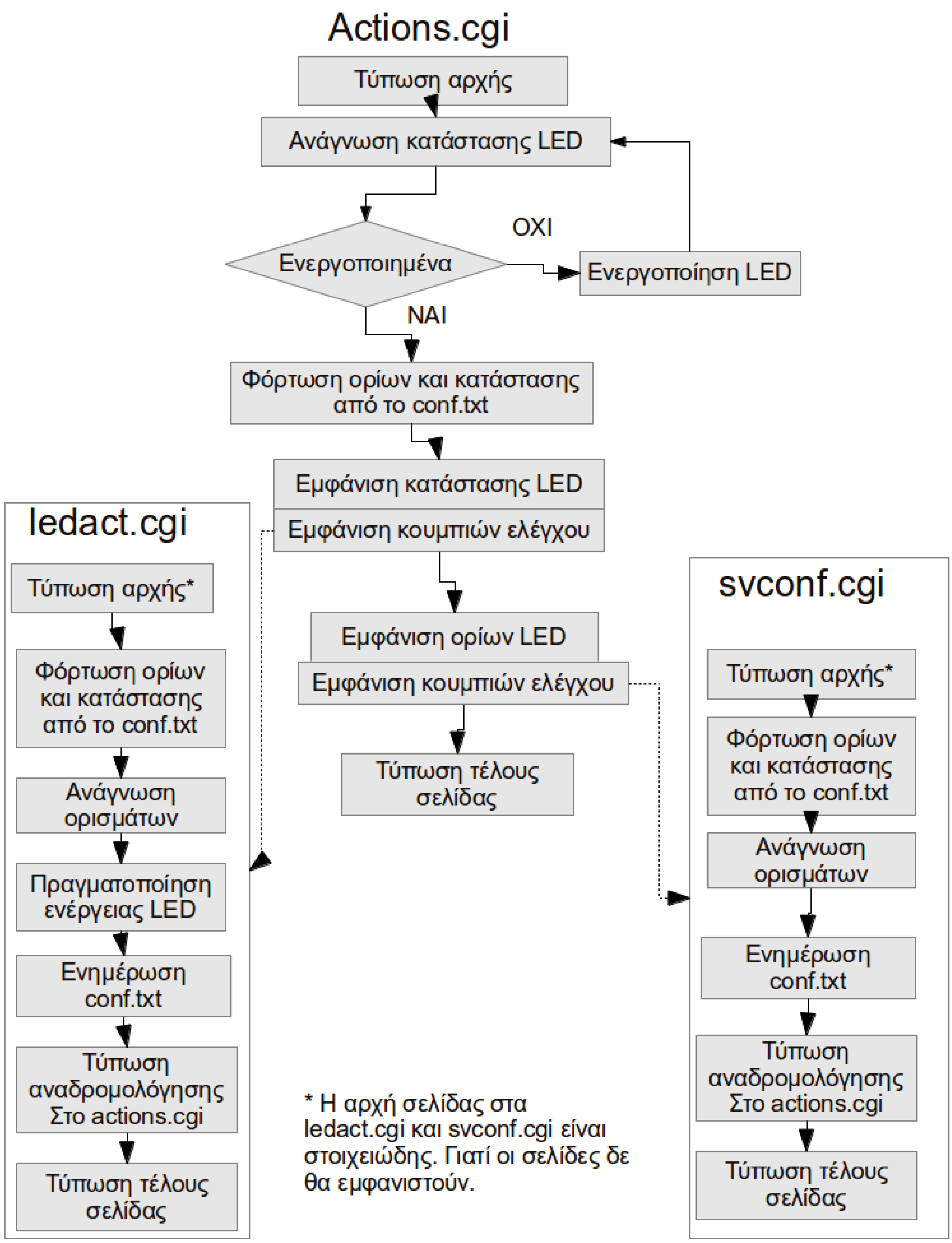


5.6.3 Actions antions.cgi

Οι, δεύτερης βαθμίδας, κουκκίδες είναι οι ενέργειες των cgi που καλούνται για να εκτελέσουν κάποιες ενέργειες. Σε αντίθεση με το metrisis.cgi οι φόρμες του actions.cgi καλούν βοηθητικά cgi τα οποία επεξεργάζονται τα ορίσματα, εκτελούν τις επιθυμητές ενέργειες και τέλος καλούν το actions.cgi, δίνοντας του και πάλι τον έλεγχο. Καθώς το actions.cgi φορτώνει εκ νέου, γίνονται ορατά και τα αποτελέσματα των ενεργειών των βοηθητικών cgi.

- Τύπωση αρχής
- Ανάγνωση κατάστασης LED. Διαβάζοντας τα αρχεία "/sys/class/gpio/gpio38/value" και "/sys/class/gpio/gpio40/value" για τα LED 1 και 2 αντίστοιχα, το πρόγραμμα καταλαβαίνει αν είναι αναμμένα ή σβηστά.
- Αν τα αρχεία δεν υπάρχουν, σημαίνει ότι τα LED (οι ακροδέκτες GPIO που αντιστοιχούν σε αυτά) δεν έχουν ενεργοποιηθεί. Το actions.cgi τα ενεργοποιεί και επαναλαμβάνει το προηγούμενο βήμα.
- Φορτώνονται τα όρια και η λειτουργία (mode) των LED από το conf.txt.
- Εμφανίζονται τα κουμπιά ελέγχου της λειτουργίας (mode) των LED. Με το πάτημα τους μπορούμε να αλλάξουμε την λειτουργία των LED σε: αναμμένο, σβηστό, αυτόματο. Προκειμένου να πραγματοποιηθεί αυτή η ενέργεια, καλείται το ledact.cgi με τα κατάλληλα ορίσματα. Αυτό αφού εκτελέσει την εντολή, καλεί ξανά το actions.cgi, όπου βλέπουμε τις αλλαγές.
    - Ledact.cgi Τύπωση αρχής. Τα βοηθητικά cgi όπως αυτό, εμφανίζονται στιγμιαία πριν ανακατευθύνουν τον έλεγχο στο cgi που τα κάλεσε. Καθώς ο χρήστης σπάνια θα τα δει και όταν τα δει (σε περίπτωση σφάλματος) χρειάζεται να αλληλεπιδράσει με αυτά έχουν μια πιο στοιχειώδη αρχή, χωρίς μενού και css.
    - Φορτώνονται τα όρια και η λειτουργία (mode) των LED από το conf.txt.
    - Ανάγνωση ορισμάτων
    - Πραγματοποιείται η ενέργεια LED. Ανάλογα με το όρισμα αλλάζει η τιμή στο αρχείο "/sys/class/gpio/gpio<LED PIN>/value"
    - Ενημερώνεται το αρχείο conf.txt. Αν αλλάξουμε σε λειτουργία ON ή OFF, εκτός από το να ανάψει το LED, καταγράφεται αυτή η επιλογή και στο αρχείο ρυθμίσεων, ώστε το envstored να μην επέμβει. Αν αλλάξουμε σε λειτουργία auto, η κατάσταση του LED μένει ως έχει (θα την αλλάξει το envstored την επόμενη μέτρηση) και το μόνο που αλλάζει είναι η καταχώρηση στο conf.txt.
    - Τυπώνεται ο κώδικας javascript που αναδρομολογεί στο actions.cgi.
    - Τύπωση τέλους σελίδας
- Εμφανίζονται τα όρια των LED. Η τιμή του κάθε ορίου τυπώνεται μέσα σε πεδίο κειμένου που επιδέχεται αλλαγή (textbox). Για να αλλάξουμε τα όρια, αρκεί να αλλάξουμε την τιμή του πεδίου και να πατήσουμε "save configuration"
    - Αυτή η ενέργεια καλεί το svconf.cgi το οποίο τυπώνει τη αρχή της σελίδας του όπως το ledact.cgi
    - Φορτώνονται τα όρια και η λειτουργία (mode) των LED από το conf.txt.
    - Ανάγνωση ορισμάτων
    - Ενημερώνεται το conf.txt. Η λειτουργία των LED παραμένει ως έχει, ενώ τα όρια αντικαθίστανται με αυτά που ελήφθησαν ως ορίσματα.
    - Τυπώνεται η αναδρομολόγηση στο actions.cgi όπως και στο ledact.cgi
    - Τύπωση τέλους σελίδας
- Τύπωση τέλους σελίδας

Η τύπωση (αρχής και τέλους σελίδας για παράδειγμα) αναφέρεται όπως είναι φυσικό στην τύπωση κώδικα ο οποίος στέλνεται στον περιηγητή ιστού του χρήστη. Ο χρήστης δεν είναι απαραίτητο να δει τη τυπώθηκε.



Actions.cgi
Τύπωση αρχής
Ανάγνωση κατάστασης LED
Ενεργοποιημένα
ΟΧΙ
Ενεργοποίηση LED
ΝΑΙ
Φόρτωση ορίων και κατάστασης από το conf.txt
Εμφάνιση κατάστασης LED
Εμφάνιση κουμπιών ελέγχου
Εμφάνιση ορίων LED
Εμφάνιση κουμπιών ελέγχου
Τύπωση τέλους σελίδας
ledact.cgi
Τύπωση αρχής*
Φόρτωση ορίων και κατάστασης από το conf.txt
Ανάγνωση ορισμάτων
Πραγματοποίηση ενέργειας LED
Ενημέρωση conf.txt
Τύπωση αναδρομολόγησης Στο actions.cgi
Τύπωση τέλους σελίδας
svconf.cgi
Τύπωση αρχής*
Φόρτωση ορίων και κατάστασης από το conf.txt
Ανάγνωση ορισμάτων
Ενημέρωση conf.txt
Τύπωση αναδρομολόγησης Στο actions.cgi
Τύπωση τέλους σελίδας
* Η αρχή σελίδας στα ledact.cgi και svconf.cgi είναι στοιχειώδης. Γιατί οι σελίδες δε θα εμφανιστούν.

Κεφάλαιο 6 – Αποτέλεσμα

6.1 Αποτελέσματα

Στα τέλος της εργασίας έχουμε ένα σύστημα μέτρησης θερμοκρασίας και υγρασίας, που κρατάει ιστορικό μετρήσεων. Δίνει τη δυνατότητα να αναζητήσουμε παλαιότερες ρυθμίσεις, με επαρκή κριτήρια ώστε να επιλέξουμε ακριβώς το δείγμα που επιθυμούμε.

Επίσης ελέγχει δύο LED. Ο έλεγχος τους μπορεί να γίνει είτε χειροκίνητα μέσα από τη διεπαφή, είτε αυτόματα από το πρόγραμμα. Στην περίπτωση που επιλέξουμε αυτόματα, ορίζουμε μέσα από τη διεπαφή, μέγιστη και ελάχιστη θερμοκρασία και υγρασία. Όταν κάποιο από τα μέγιστα ξεπεραστεί ανάβει το LED1 και όταν κάποιο από τα ελάχιστα ξεπεραστεί προς τα κάτω, ανάβει το LED2.

Το σκεπτικό είναι ότι αν οι ακροδέκτες των LED συνδεθούν με άλλες συσκευές, όπως ένα σύστημα θέρμανσης, έναν ανεμιστήρα, ένα κλιματιστικό, η εφαρμογή αυτή θα γίνει ένα σύστημα εποπτείας και ελέγχου θερμοκρασίας και υγρασίας. Η αυτόματη λειτουργία θα μπορούσε να διατηρήσει το χώρο σε κατάλληλες συνθήκες, ενώ η χειροκίνητη λειτουργία θα επέτρεπε την παράκαμψη του αυτόματου συστήματος για μια έκτακτη αλλαγή των συνθηκών αν αυτό κριθεί απαραίτητο.

Σε μια τέτοια περίπτωση, καλό θα ήταν να τοποθετηθεί η διεργασία που λαμβάνει τις μετρήσεις και πραγματοποιεί τις ενέργειες της αυτόματης λειτουργίας (envstored), σε αυτόματη εκτέλεση. Ώστε να εκκινεί αυτόματα, μετά από πιθανή διακοπή τροφοδοσίας.

Επιπλέον η διεπαφή είναι σε μορφή ιστοσελίδας και επομένως είναι προσβάσιμη σχεδόν από οποιαδήποτε συσκευή στο ίδιο δίκτυο. Με την κατάλληλη σύνδεση, θα μπορούσε βέβαια να είναι προσβάσιμη στο διαδίκτυο, ώστε να έχουμε έλεγχο του χώρου από οπουδήποτε.

6.2 Συμπεράσματα

Με τη χρήση μίας πλακέτας με ARM μικροελεγκτή, μπορούμε να δημιουργήσουμε ένα σύστημα εποπτείας και ελέγχου περιβαλλοντικών μεταβλητών. Μπορούμε επίσης να παρέχουμε διεπαφή βασισμένη σε ιστοσελίδα και να αποθηκεύουμε τα δεδομένα του αισθητήρα σε βάση δεδομένων.

Σε αυτή την προσπάθεια βοηθάνε πολύ οι διανομές Linux για ενσωματωμένα συστήματα. Είναι τόσο πλήρεις από άποψη εφαρμογών στα αποθετήρια τους, που πλέων ένα ενσωματωμένο σύστημα μπορεί να κάνει σχεδόν ότι και ένας υπολογιστής. Ο κυριότερος φραγμός είναι οι επιδώσεις του ενσωματωμένου συστήματος, οι οποίες είναι μεν σαφώς χαμηλότερες από αυτές ενός προσωπικού υπολογιστή, αλλά μπορούν να συγκριθούν με προσωπικούς υπολογιστές περασμένης τεχνολογίας.

Πολλά από τα προβλήματα που έπρεπε να αντιμετωπιστούν βέβαια, αφορούσαν έλλειψη υποστήριξης για το συγκεκριμένο υλικό. Επειδή τα ενσωματωμένα συστήματα έχουν πολλές διαφορές μεταξύ τους, και οι άνθρωποι που ασχολούνται με αυτά (με τον προγραμματισμό και την παραμετροποίηση τους) είναι λιγότεροι από όσους ασχολούνται με συμβατούς υπολογιστές, είναι πιο δύσκολη η ανεύρεση λογισμικού ή οδηγιών για συγκεκριμένο υλικό. Καλό θα ήταν, αν κάποιος θέλει να ασχοληθεί, ειδικά αν δεν έχει πείρα, να επιλέξει μια δημοφιλή πλακέτα, με ενεργή κοινότητα.

Η mini2440 είναι λίγο παλιά, η κοινότητα της δεν είναι ιδιαιτέρα ενεργή, και πολλά πακέτα της Angstrom δεν είναι διαθέσιμα για το ARMv4 σετ εντολών του επεξεργαστή της, ή είναι διαθέσιμες μόνο παλιές εκδώσεις. Επιπλέων η εταιρεία ανάπτυξης δεν εξέδωσε καν ένα αξιοπρεπές εγχειρίδιο στα αγγλικά. Πιθανότατα το Raspberry Pi είναι μια καλή επιλογή.

6.3 Μελλοντική εργασία και επεκτάσεις

Η εργασία έχει πολλές δυνατότητες επέκτασης. Μια βελτίωση θα ήταν η προσθήκη περισσότερων αισθητήρων. Όπως αναφέρθηκε στην ανάλυση του πρωτοκόλλου SPI (4.11 SPI), υπάρχει η δυνατότητα να χρησιμοποιήσουμε ακροδέκτες της GPIO ως CS. Οι περισσότεροι αισθητήρες θα ήταν κατάλληλοι για την εποπτεία μεγαλύτερων χώρων ή πολλών χώρων με ένα μόνο σύστημα. Δια παράδειγμα θα μπορούσε να ελέγχει ένα μικρό συγκρότημα θερμοκηπίων, ελέγχοντας εξαερισμό και θέρμανση. Ένα βήμα πιο πέρα θα ήταν να τροποποιηθεί το λογισμικό για να προσαρμόζεται δυναμικά στον αριθμό τον συνδεδεμένων αισθητήρων, κάνοντας πιο εύκολη την προσθήκη ή αφαίρεση τους από μη εξειδικευμένο χρήστη. Άλλοι αισθητήρες όπως μετρητές οξύτητας θα μπορούσαν να

συνδεθούν επίσης για να καλύψουν διαφορετικές ανάγκες ελέγχου.

Σημαντική βελτίωση θα αποτελούσε η προσθήκη οθόνης αφής και λειτουργικού γραφικού περιβάλλοντος στην πλακέτα. Θα μπορούσε να τρέχει μια διεπαφή παρόμοια με αυτή που τρέχει σε web server. Έτσι η πλακέτα θα ήταν πλήρως αυτονομημένη. Η mini2440 έχει τέτοια δυνατότητα αλλά δε χρησιμοποιήθηκε.

Ένα βήμα πιο πέρα θα ήταν η προσθήκη κάμερας. Εκτός από την ύπαρξη USB, κυκλοφορεί κάμερα ειδικά για την mini2440. Αυτό θα έδινε και οπτική εποπτεία του χώρου. Τέλος υπάρχει ασύρματη κάρτα δικτύου για τη mini2440. Αυτό θα επέτρεπε την τοποθέτηση της σε δύσκολα προσβάσιμα με δεδομένη τη δυνατότητα δικτύωσης.

Βιβλιογραφία

(1) MicroArm Systems , “MINI2440 User’s Manual ” 2009-03-03
(MINI2440_USER_MANUAL.pdf) (http://www.friendlyarm.net/downloads)
(2) (LinuxMCE wiki) Compil_install_Uboot_Kernel_HowTo.pdf
(3) Sure Electronics Inc “Temperature and Relative Humidity Sensor Module User’s Guide ”
DC-SS500_Ver1.0
(4) FriendlyARM mini2440 (mini2440_manual.pdf)
(5) Karim Yaghmour , “Building Embedded Linux Systems ”
(Building.Embedded.Linux.Systems.pdf)
(6) Michael Barr & Antony Massa, “Programming Embedded Systems with C and GNU
Developement tools” (http://books.google.gr/books?
id=nPZaPJrw_L0C&pg=PA1&redir_esc=y#v=onepage&q&f=true)
(7) Steve,Heath, “Embedded Systems Design” (http://books.google.gr/books?
id=BjNZXwH7HlkC&pg=PA2&redir_esc=y#v=onepage&q&f=false)

Διαδίκτυο
(8) Bill's Mini2440 Forum
http://billforums.station51.net/index.php

- HOWTO: Compiling and Flashing U-Boot onto Mini2440
- http://billforums.station51.net/viewtopic.php?f=1&t=2
- HOWTO: Booting from NFS using U-Boot
- http://billforums.station51.net/viewtopic.php?f=1&t=17
- HOWTO: Booting from SD card using U-Boot
- http://billforums.station51.net/viewtopic.php?f=1&t=5
- HOWTO: Getting Started With OpenEmbedded
- http://billforums.station51.net/viewtopic.php?f=3&t=4
- HOWTO: Cross compiling the kernel for the Mini2440
- http://billforums.station51.net/viewtopic.php?f=4&t=20

(9) Emeb, FriendlyARM mini2440
http://members.cox.net/ebrombaugh1/embedded/mini2440/index.html
(μη διαθέσιμη, παρέχεται αντίγραφο στο CD)
(10) FriendlyARM official site
http://www.friendlyarm.net/
(11) FriendlyARM official forum
http://www.friendlyarm.net/forum
http://www.friendlyarm.net/forum/topic/243
(12) Wikipedia
https://en.wikipedia.org/wiki/Wiki

- http://en.wikipedia.org/wiki/GPIO
- http://en.wikipedia.org/wiki/Sysfs
- https://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus
- http://en.wikipedia.org/wiki/Sqlite
- http://en.wikipedia.org/wiki/OpenEmbedded
- http://en.wikipedia.org/wiki/Reduced_instruction_set_computing
- http://en.wikipedia.org/wiki/ARM_architecture

(13) Accessing SQLite in C
http://www.linuxjournal.com/content/accessing-sqlite-c
(14) SQLite official site

http://www.sqlite.org/
(15) Installing sqlite headers on ubuntu
http://databasically.com/2010/03/05/installing-sqlite-headers-on-ubuntu-sqlite3-h-not-found/
(16) How to Daemonize in Linux
http://www-theorie.physik.unizh.ch/~dpotter/howto/daemonize?do=show
(17) How to create a single instance application in C
http://stackoverflow.com/questions/5339200/how-to-create-a-single-instance-application-in-c
(18) Documentation:Linux/GPIO
http://www.avrfreaks.net/wiki/index.php/Documentation:Linux/GPIO
(19) How to Daemonize in Linux
http://www-theorie.physik.unizh.ch/~dpotter/howto/daemonize?do=show
(20) Footnotes for "CGI Made Really Easy"
http://www.jmarshall.com/easy/cgi/

Κώδικες

Ανάγνωση ορισμάτων get:
Παράγει ζεύγη τύπου πεδίο – τιμή, των δεδομένων που ελήφθησαν με get.

```
/** Convert a two-char hex string into the char it represents. **/
char x2c(char *what) {
	register char digit;

	digit = (what[0] >= 'A' ? ((what[0] & 0xdf) - 'A')+10 :
(what[0] - '0'));
	digit *= 16;
	digit += (what[1] >= 'A' ? ((what[1] & 0xdf) - 'A')+10 :
(what[1] - '0'));
	return(digit);
}

/** Reduce any %xx escape sequences to the characters they
represent. **/
void unescape_url(char *url) {
register int i,j;

	for(i=0,j=0; url[j]; ++i,++j) {
		if((url[i] = url[j]) == '%') {
			url[i] = x2c(&url[j+1]) ;
			j+= 2 ;
		}
	}
	url[i] = '\0' ;
}

/** Read the CGI input and place all name/val pairs into list.
	**/
/** Returns list containing name1, value1, name2, value2, ... , NULL
**/
char **getcgivars() {
	register int i ;
	char *request_method ;
	int content_length;
	char *cgiinput ;
	char **cgivars ;
	char **pairlist ;
	int paircount ;
	char *nvpair ;
	char *eqpos ;

	/** Depending on the request method, read all CGI input into
cgiinput. **/
	request_method= getenv("REQUEST_METHOD") ;

	if (!strcmp(request_method, "GET") || !strcmp(request_method,
"HEAD") ) {		//strcmp=string_compare
		/* Some servers apparently don't provide QUERY_STRING if
it's empty, */
```

```
            /*   so avoid strdup()'ing a NULL pointer here.    */
            char *qs ;
            qs= getenv("QUERY_STRING") ;
            cgiinput= strdup(qs  ? qs  : "") ;
      }
      else if (!strcmp(request_method, "POST")) {
            /* strcasecmp() is not supported in Windows-- use
strcmpi() instead */
            if ( strcasecmp(getenv("CONTENT_TYPE"),
"application/x-www-form-urlencoded")) {
      //strcmp=string_compare_ingorecase
            printf("Content-Type: text/plain\n\n") ;
                  printf("getcgivars(): Unsupported Content-Type.\n")
;
                  exit(1) ;
            }
            if ( !(content_length = atoi(getenv("CONTENT_LENGTH"))) )
{
            printf("Content-Type: text/plain\n\n") ;
                  printf("getcgivars(): No Content-Length was sent
with the POST request.\n") ;
                  exit(1) ;
            }
            if ( !(cgiinput= (char *) malloc(content_length+1)) ) {
            printf("Content-Type: text/plain\n\n") ;
                  printf("getcgivars(): Couldn't malloc for
cgiinput.\n") ;
                  exit(1) ;
            }
            if (!fread(cgiinput, content_length, 1, stdin)) {
            printf("Content-Type: text/plain\n\n") ;
                  printf("getcgivars(): Couldn't read CGI input from
STDIN.\n") ;
                  exit(1) ;
            }
            cgiinput[content_length]='\0' ;
      }
      else {
      printf("Content-Type: text/plain\n\n") ;
            printf("getcgivars(): Unsupported REQUEST_METHOD.\n") ;
            exit(1) ;
      }

      /** Change all plusses back to spaces. **/
      for (i=0; cgiinput[i]; i++) if (cgiinput[i] == '+') cgiinput[i]
= ' ' ;
      /** First, split on "&" and ";" to extract the name-value pairs
into **/
      /**   pairlist.
                          **/
      pairlist= (char **) malloc(256*sizeof(char **)) ;
      paircount= 0 ;
      nvpair= strtok(cgiinput, "&;") ;
      while (nvpair) {
            pairlist[paircount++]= strdup(nvpair) ;
```

```
		if (!(paircount%256))
			pairlist= (char **) realloc(pairlist,
(paircount+256)*sizeof(char **)) ;
		nvpair= strtok(NULL, "&;") ;
	}
	pairlist[paircount]= 0 ;   /* terminate the list with NULL */

	/** Then, from the list of pairs, extract the names and values.
**/
	cgivars= (char **) malloc((paircount*2+1)*sizeof(char **)) ;
	for (i= 0; i<paircount; i++) {
		if (eqpos=strchr(pairlist[i], '=')) {
			*eqpos= '\0' ;
			unescape_url(cgivars[i*2+1]= strdup(eqpos+1)) ;
		} else {
			unescape_url(cgivars[i*2+1]= strdup("")) ;
		}
		unescape_url(cgivars[i*2]= strdup(pairlist[i])) ;
	}
	cgivars[paircount*2]= 0 ;   /* terminate the list with NULL */
	                           //seepoint
	/** Free anything that needs to be freed. **/
	free(cgiinput) ;
	for (i=0; pairlist[i]; i++) free(pairlist[i]) ;
	free(pairlist) ;

	/** Return the list of name-value strings. **/
	return cgivars ;
}
```

Μεταφορά στο παρασκήνιο:

```
static void daemonize(const char *lockfile)
{
	pid_t pid, sid, pid_file, rc;

	/* already a daemon */
	if ( getppid() == 1 ) return;

	/* Create the lock file as the current user */
	pid_file = open(lockfile,O_RDWR|O_CREAT,0640);
	rc = flock(pid_file, LOCK_EX | LOCK_NB);
	if(rc) {
		if(EWOULDBLOCK == errno){
			puts("Already runnung");
			exit(EXIT_FAILURE);
		}
	}

	/* Fork off the parent process */
	pid = fork();
	if (pid < 0) {
		exit(EXIT_FAILURE);
	}
```

```
	/* If we got a good PID, then we can exit the parent process.
*/
	if (pid > 0) {
		exit(EXIT_SUCCESS);
	}

	/* At this point we are executing as the child process */

	/* Change the file mode mask */
	umask(0);

	/* Create a new SID for the child process */
	sid = setsid();
	if (sid < 0) {
		exit(EXIT_FAILURE);
	}

	/* Redirect standard files to /dev/null */
	freopen( "/dev/null", "r", stdin);
	freopen( "/dev/null", "w", stdout);
	freopen( "/dev/null", "w", stderr);
}
```

και μέσα στη main:

```
/*Call deamonize function to run as a deamon*/
	if (deamonic == 1){
		printf("Deamon mode: run in background\n");
		daemonize( "/var/lock/subsys/" DAEMON_NAME );
	}
	else{
		printf("Angel mode: output on terminal for problem
detection\n");
}
```

Ανάγνωση ορισμάτων γραμμής εντολών:

Διαβάζει τα ορίσματα του envstored και εκτελεί την ανάλογη ενέργεια.

```
static void parse_opts(int argc, char *argv[])
{
	while (1) {
		static const struct option lopts[] = {
			{ "scrn",	0, 0, 's' },
			{ "help",	0, 0, 'h' },
			{ NULL, 0, 0, 0 },
		};
		int c;

		c = getopt_long(argc, argv, "sh", lopts, NULL);

		if (c == -1)
			break;

		switch (c) {
		case 's':
```

```
                deamonic=0;
                break;
            case 'h':
                puts("envstored help!\n"
                " Kalampoukas Xristos A.M.1024\n"
                "  -s --scrn  print output (to detect problems)\n"
                "  -h --help  print this\n");
                exit(1);
                break;
            default:
                printf("Deamon mode\n");
                break;
            }
        }
}
```

Ανταλλαγή δεδομένων με αισθητήρα:
Η ανάγνωση δεδομένων τελειώνει με το τέλος της do – while. Ο υπόλοιπος κώδικας από εκεί και κάτω καλεί της συναρτήσεις αποθήκευσης στη βάση δεδομένων και ενεργοποίησης – απενεργοποίησης των LED αν αυτό είναι απαραίτητο.

```
static void transfer(int fd)
{
        int ret,c=0,ther=0,hyd=0;
        uint8_t tx[] = {0xa0};
        int maxt,mint,maxh,minh,led1mod = 0,led2mod = 0;
        time_t now;


        uint8_t rx[ARRAY_SIZE(tx)] = {0, };
        struct spi_ioc_transfer tr = {
                .tx_buf = (unsigned long)tx,
                .rx_buf = (unsigned long)rx,
                .len = ARRAY_SIZE(tx),
                .delay_usecs = delay,
                .speed_hz = speed,
                .bits_per_word = bits,
                .cs_change = 1,
        };

        /*Lipsi metrisewn*/
        do{
                c++;

                ret = ioctl(fd, SPI_IOC_MESSAGE(1), &tr);
                if (ret < 1){
                        pabort("can't send spi message");
                }

                if(c==2) hyd=(int)rx[0];
                if(c==3) ther=(int)rx[0];
                printf("Esteile %X kai pire %d\n", tx[0],rx[0]);
```

```
            tx[0]=tx[0]+16;

      }while(c<3);

      /*Tupwsi hmerominias kai metrisis an den trexei sto 
paraskinio*/
      if (deamonic == 0){
            time(&now);
            printf("%s", ctime(&now));
            printf("H thermokrasia einai: %d\nH ugrasia einai: 
%d\n",ther,hyd);
      }

      /*store new variables in database if diferent from the old 
ones*/
      if ((ther != last_heat) || (hyd != last_humidity)){
            store(ther,hyd);
            last_heat = ther;
            last_humidity = hyd;
      }

      load_limits(&maxt, &mint, &maxh, &minh, &led1mod, &led2mod);

      if(((ther > maxt) || (hyd > maxh)) && (led1mod == 0)){
            led_act("LED1_ON");
      }
      else if (led1mod == 0){
            led_act("LED1_OFF");
      }
      if(((ther < mint) || (hyd < minh)) && (led2mod == 0)){
            led_act("LED2_ON");
      }
      else if (led2mod == 0){
            led_act("LED2_OFF");
      }

      printf("------------------------------------\n");
}
```

Χείρισμός LEDS:
Η init_leds ελέγχει αν τα LED είναι ενεργοποιημένα, αν όχι τα ενεργοποιεί. Η led_act ανάβει ή σβήνει ένα LED ανάλογα με το όρισμα που δέχετε.

```
/*Do the led action*/
led_act(char *act){
      if (strcmp(act, "LED1_ON") == 0){
            system("echo 0 > /sys/class/gpio/gpio38/value");
      }
      else if (strcmp(act, "LED2_ON") == 0){
            system("echo 0 > /sys/class/gpio/gpio40/value");
      }
      else if (strcmp(act, "LED1_OFF") == 0){
```

```
			system("echo 1 > /sys/class/gpio/gpio38/value");
		}
		else if (strcmp(act, "LED2_OFF") == 0){
			system("echo 1 > /sys/class/gpio/gpio40/value");
		}
}

/*Initialize leds*/
int init_leds(){
		int status;
		int ret = 0;
		FILE *istream;

		/*Enable LED1*/
		if ( (istream = fopen ( "/sys/class/gpio/gpio38/direction", "r"
) ) == NULL ){
			status = system("echo 38 > /sys/class/gpio/export");
			if(status!=0){
				puts("Problem enabling LED1");
				ret = 1;
			}
			status = system("echo out >
/sys/class/gpio/gpio38/direction");
			if(status!=0){
				puts("Problem enabling LED1");
				ret = 1;
			}
		}
		else{
			printf ( "LED1 already enabled\n" );
			fclose ( istream );
		}

		/*Enable LED2*/
		if ( (istream = fopen ( "/sys/class/gpio/gpio40/direction", "r"
) ) == NULL ){
			status = system("echo 40 > /sys/class/gpio/export");
			if(status!=0){
				puts("Problem enabling LED2");
				ret = 1;
			}
			status = system("echo out >
/sys/class/gpio/gpio40/direction");
			if(status!=0){
				puts("Problem enabling LED2");
				ret = 1;
			}
		}
		else{
			printf ( "LED2 already enabled\n" );
			fclose ( istream );
		}
```

```
	return ret;
}
```

Φόρτωση μεταβλητών από conf.txt:

```
/*Load conf vatiables*/
file = fopen (filename, "r");
if ( file != NULL ){
	while ( fgets ( line, sizeof line, file ) != NULL ){ /* read a
line */
		if(strncmp(line,"max tempetarure",14) == 0){
			value = strchr(line,'\t');
			maxt = atoi(value);
		}
		if(strncmp(line,"max humidity",12) == 0){
			value = strchr(line,'\t');
			maxh = atoi(value);
		}
		if(strncmp(line,"min temperature",13) == 0){
			value = strchr(line,'\t');
			mint = atoi(value);
		}
		if(strncmp(line,"min humitity",11) == 0){
			value = strchr(line,'\t');
			minh = atoi(value);
		}
		if(strncmp(line,"LED1",4) == 0){
			value = strchr(line,'\t');
			led1mod = atoi(value);
		}
		if(strncmp(line,"LED2",4) == 0){
			value = strchr(line,'\t');
			led2mod = atoi(value);
		}
		i++;
	}
	fclose (file);
}
else{
	perror (filename); /* why didn't the file open? */
}
/*Telos Load conf vatiables*/
```

Αποθήκευση δεδομένων στο conf.txt:

```
/*write to file*/
fprintf(file,"max tempetarure\t%d\n",maxt);
fprintf(file,"max humidity\t%d\n",maxh);
fprintf(file,"min temperature\t%d\n",mint);
fprintf(file,"min humitity\t%d\n",minh);
fprintf(file,"LED1\t%d\n",led1mod);
fprintf(file,"LED2\t%d\n",led2mod);
```

Αποθήκευση μετρήσεων στη βάση δεδομένων:

Χρησιμοποιείται στο πρόγραμμα envstored.

```
static void store (int ther, int hyd)
{
	sqlite3 *conn;			//database connection pionter
	int	error = 0;
	char *zsql;			//Querry
	char *errmsg;			//sqlite_exec possible error
	char si[10], ni[10];		//strings for our integers

	/*Connect to database*/
	error = sqlite3_open("envdata.sl3", &conn);
	if (error) {
		puts("Can not open database");
		exit(0);
	}


	sprintf(si,"%d",ther);			//int to string
	sprintf(ni,"%d",hyd);
	zsql = sqlite3_mprintf(
		"insert into envstore (time, heat, humidity) values
(datetime('now'), '%q', '%q')"
		, si, ni);
	error = sqlite3_exec(conn,zsql,0, 0, &errmsg);
		//execute querry
	sqlite3_free(zsql);
	//free querry memory
	if (errmsg != NULL) puts(errmsg);
		//prints sqlite_exec error if exists
	sqlite3_free(errmsg);
	//free memory

	if (error != SQLITE_OK) {
		puts("Database related error");
		exit(0);
	}

	printf("data saved\n");

	/*katharisma*/
	sqlite3_close(conn);		//free database memory
}
```

Δείγμα κανονικοποίησης ημερομηνίας και ώρας:

Συμπληρώνει τα ψηφία που λείπουν από την ημερομηνία και την ώρα που πληκτρολόγησε ο χρήστης.

```
/*from*/
if ((strcmp(dfrom, "null") != 0) || (strcmp(hfrom, "null") != 0)){
	/*Dfrom*/
	if (strcmp(dfrom, "null") == 0){
		strncat(strtmp2,tdate,10);
		sprintf(dfrom,"%s",strtmp2);
	}
	else if (dfrom[2] == '-'){
		strcpy(strtmp1,dfrom);
```

```
        strncat(strtmp2,tdate,4);
        strtmp2[4] = '\0';
        sprintf(dfrom,"%s-%s",strtmp2,strtmp1);
    }

    /*Hfrom*/
    if (strcmp(hfrom, "null") == 0){
        strcpy(hfrom,"00:00:00");
    }else if (strlen(hfrom) == 5){
        strcat(hfrom,":00");
    }else if (strlen(hfrom) == 2){
        strcat(hfrom,":00:00");
    }
}

strcpy(from,dfrom);
strcat(from," ");
strcat(from,hfrom);
/*telos from*/
```

Τύπωση αποτελεσμάτων:

Χρησιμοποιείται στο πρόγραμμα metrisis.cgi

```
/*Arxikopoiisi*/
sprintf(scount, "0");
sprintf(count, "0");
strcpy(dfrom,"null");
strcpy(dto,"null");
strcpy(hfrom,"null");
strcpy(hto,"null");
strcpy(sqlqtmp,"null");

/*Anagnwrisi orismatwn*/
for(i=0; cgivars[i]; i++){
    if (strcmp(cgivars[i], "sel") == 0){
        sel=atoi(cgivars[i+1]);
    }
    if (strcmp(cgivars[i], "nai") == 0){
        ola=1;
    }
    if (strcmp(cgivars[i], "dfrom") == 0){
        if (strcmp(cgivars[i+1], "YYYY-MM-DD") != 0){
            strcpy(dfrom,cgivars[i+1]);
            krit = 1;
        }
    }
    if (strcmp(cgivars[i], "dto") == 0){
        if (strcmp(cgivars[i+1], "YYYY-MM-DD") != 0){
            strcpy(dto,cgivars[i+1]);
            krit = 1;
        }
    }
    if (strcmp(cgivars[i], "hfrom") == 0){
        if (strcmp(cgivars[i+1], "hh:mm:ss") != 0){
```

```
                strcpy(hfrom,cgivars[i+1]);
                krit = 1;
            }
        }
        if (strcmp(cgivars[i], "hto") == 0){
            if (strcmp(cgivars[i+1], "hh:mm:ss") != 0){
                strcpy(hto,cgivars[i+1]);
                krit = 1;
            }
        }
        if (strcmp(cgivars[i], "maxt") == 0){
            if (strcmp(cgivars[i+1], "-") != 0){
                maxt=atoi(cgivars[i+1]);
                krit = 1;
            }
        }
        if (strcmp(cgivars[i], "maxh") == 0){
            if (strcmp(cgivars[i+1], "-") != 0){
                maxh=atoi(cgivars[i+1]);
                krit = 1;
            }
        }
        if (strcmp(cgivars[i], "mint") == 0){
            if (strcmp(cgivars[i+1], "-") != 0){
                mint=atoi(cgivars[i+1]);
                krit = 1;
            }
        }
        if (strcmp(cgivars[i], "minh") == 0){
            if (strcmp(cgivars[i+1], "-") != 0){
                minh=atoi(cgivars[i+1]);
                krit = 1;
            }
        }
    }

    /*kanonikopoiisi from to*/
    if(krit == 1)
        formarg(dfrom,dto,hfrom,hto,from,to);

    /*kataskeui querry kritiriwn*/
    if (from[4] == '-'){
        if(strcmp(sqlqtmp, "null") == 0){
            sprintf(sqlqtmp,"time >= datetime('%s')",from);
        }
        else{
            sprintf(tempstring," and time >= datetime('%s')",from);
            strcat(sqlqtmp,tempstring);
        }
    }
    if (to[4] == '-'){
        if(strcmp(sqlqtmp, "null") == 0){
            sprintf(sqlqtmp,"time <= datetime('%s')",to);
```

```
        }
        else{
            sprintf(tempstring," and time <= datetime('%s')",to);
            strcat(sqlqtmp,tempstring);
        }
}
if (maxt != -256){
        if(strcmp(sqlqtmp, "null") == 0){
            sprintf(sqlqtmp,"heat <= %d",maxt);
        }
        else{
            sprintf(tempstring," and heat <= %d",maxt);
            strcat(sqlqtmp,tempstring);
        }
}
if (maxh != -256){
        if(strcmp(sqlqtmp, "null") == 0){
            sprintf(sqlqtmp,"humidity <= %d",maxh);
        }
        else{
            sprintf(tempstring," and humidity <= %d",maxh);
            strcat(sqlqtmp,tempstring);
        }
}
if (mint != -256){
        if(strcmp(sqlqtmp, "null") == 0){
            sprintf(sqlqtmp,"heat >= %d",mint);
        }
        else{
            sprintf(tempstring," and heat >= %d",mint);
            strcat(sqlqtmp,tempstring);
        }
}
if (minh != -256){
        if(strcmp(sqlqtmp, "null") == 0){
            sprintf(sqlqtmp,"humidity >= %d",minh);
        }
        else{
            sprintf(tempstring," and humidity >= %d",minh);
            strcat(sqlqtmp,tempstring);
        }
}

/*Anigma basis*/
error = sqlite3_open("/my/envdata.sl3", &conn);
if (error) {
        system("date >> /my/sel_log.txt");
        system("echo Can not open database >> /my/sel_log.txt");
        exit(0);
}

/*epilogi drasis*/
if(cgivars[0]!= NULL){
```

```
	if(ola==1){
		/*Proetoimasia query ola*/
		sprintf(sqlq,"select time, heat, humidity from envstore
order by time desc limit %d,20",sel*20);
		error = sqlite3_prepare_v2(conn,sqlq,1000, &res, &tail);
		sprintf(scount,"select count(*) from envstore");

		sprintf(sqliq,"select AVG(\"heat\") from envstore");
		average_heat = Qtoi(sqliq,conn);
		sprintf(sqliq,"select max(\"heat\") from envstore");
		max_heat = Qtoi(sqliq,conn);
		sprintf(sqliq,"select min(\"heat\") from envstore");
		min_heat = Qtoi(sqliq,conn);
		sprintf(sqliq,"select AVG(\"humidity\") from envstore");
		average_hym = Qtoi(sqliq,conn);
		sprintf(sqliq,"select max(\"humidity\") from envstore");
		max_hym = Qtoi(sqliq,conn);
		sprintf(sqliq,"select min(\"humidity\") from envstore");
		min_hym = Qtoi(sqliq,conn);
	}
	else if(krit == 1){
		/*Proetoimasia query kritiriwn*/
		sprintf(sqlq,"select * from envstore where %s order by
time desc limit %d,20",sqlqtmp,sel*20);
		error = sqlite3_prepare_v2(conn,sqlq,1000, &res, &tail);
		sprintf(scount,"select count(*) from envstore where
%s",sqlqtmp);

		sprintf(sqliq,"select AVG(\"heat\") from envstore where
%s",sqlqtmp);
		average_heat = Qtoi(sqliq,conn);
		sprintf(sqliq,"select max(\"heat\") from envstore where
%s",sqlqtmp);
		max_heat = Qtoi(sqliq,conn);
		sprintf(sqliq,"select min(\"heat\") from envstore where
%s",sqlqtmp);
		min_heat = Qtoi(sqliq,conn);
		sprintf(sqliq,"select AVG(\"humidity\") from envstore
where %s",sqlqtmp);
		average_hym = Qtoi(sqliq,conn);
		sprintf(sqliq,"select max(\"humidity\") from envstore
where %s",sqlqtmp);
		max_hym = Qtoi(sqliq,conn);
		sprintf(sqliq,"select min(\"humidity\") from envstore
where %s",sqlqtmp);
		min_hym = Qtoi(sqliq,conn);
	/*Proetoimasia query 10 teleutaia*/
	}
	else{
		/*an kati den paei kala me ta argument*/
		system("date >> /my/sel_log.txt");
		system("Argument related problem >> /my/sel_log.txt");
	}
```

```
}
else{
      /*Proetoimasia query 10 teleutaia*/
      error = sqlite3_prepare_v2(conn,"select time, heat, humidity
from envstore order by rowid desc limit 10",1000, &res, &tail);

      sprintf(sqliq,"select avg(x.heat) from (select t.heat from
envstore t order by rowid desc limit 10) x");
      average_heat = Qtoi(sqliq,conn);
      sprintf(sqliq,"select max(x.heat) from (select t.heat from
envstore t order by rowid desc limit 10) x");
      max_heat = Qtoi(sqliq,conn);
      sprintf(sqliq,"select min(x.heat) from (select t.heat from
envstore t order by rowid desc limit 10) x");
      min_heat = Qtoi(sqliq,conn);
      sprintf(sqliq,"select avg(x.humidity) from (select t.humidity
from envstore t order by rowid desc limit 10) x");
      average_hym = Qtoi(sqliq,conn);
      sprintf(sqliq,"select max(x.humidity) from (select t.humidity
from envstore t order by rowid desc limit 10) x");
      max_hym = Qtoi(sqliq,conn);
      sprintf(sqliq,"select min(x.humidity) from (select t.humidity
from envstore t order by rowid desc limit 10) x");
      min_hym = Qtoi(sqliq,conn);
}

/*Elegxos gia lathos me ti sqlite*/
if (error != SQLITE_OK) {
      system("date >> /my/sel_log.txt");
      system("echo We did not get any data! >> /my/sel_log.txt");
}

/*Sunoptikos pinakas*/
puts("<div class=\"flo\"><table>");
printf("<tr><td>Μέσος όρος θερμοκρασίας</td><td>%d
C</td></tr>\n",average_heat);
printf("<tr><td>Μέγιστη θερμοκρασία</td><td>%d
C</td></tr>\n",max_heat);
printf("<tr><td>Ελάχιστη θερμοκρασία</td><td>%d
C</td></tr>\n",min_heat);
printf("<tr><td>Μέσος όρος υγρασίας</td><td>%d \
%</td></tr>\n",average_hym);
printf("<tr><td>Μέγιστη υγρασία</td><td>%d \%</td></tr>\n",max_hym);
printf("<tr><td>Ελάχιστη υγρασία</td><td>%d \
%</td></tr>\n",min_hym);
puts("</table></div>");

/*Ektupwsi pinaka apotelesmatwn*/
puts("<div><table class=\"result\">");
puts("<tr><td>Χρόνος</td><td>Θερμοκρασία</td><td>Υγρασία</tr>");

while (sqlite3_step(res) == SQLITE_ROW) {
      strncpy(ttime,sqlite3_column_text(res, 0),20);
```

```
      strncpy(theat,sqlite3_column_text(res, 1),3);
      strncpy(thum,sqlite3_column_text(res, 2),3);

      printf("<tr><td>%s </td><td> %s </td><td>
%s</tr>\n",ttime,theat,thum);

      rec_count++;
}

puts("</table>");

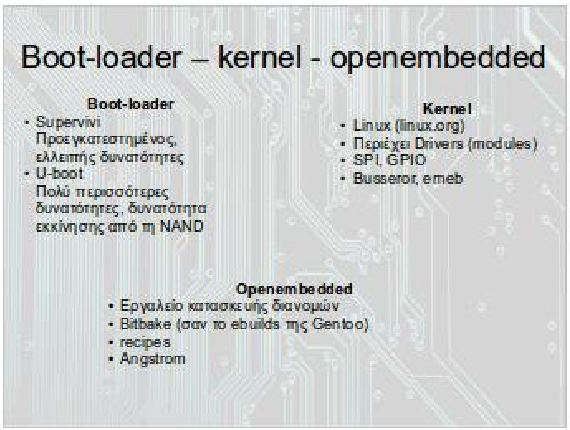
if(strcmp(scount, "0") != 0){
      /*count results*/
      error = sqlite3_prepare_v2(conn,scount,1000, &res, &tail);
      sqlite3_step(res);
      strncpy(count,sqlite3_column_text(res, 0),5);
      if (error != SQLITE_OK) {
            system("date >> /my/sel_log.txt");
            system("echo We did not get any data! >>
/my/sel_log.txt");
      }
}
```

Παρουσίαση

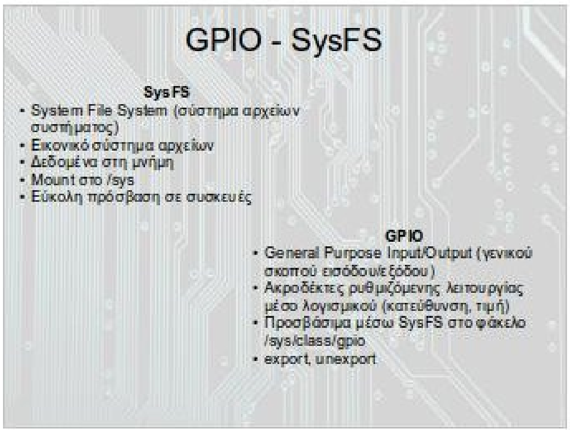

Αυτόνομο σύστημα παρακολούθησης και ελέγχου περιβαλλοντικών συνθηκών βασισμένο στην πλατφόρμα ARM mini2440
Καλαμπούκας Χρήστος Α.Μ. 1024
Επιβλέπων καθηγητής : Γεώργιος Κορνάρος
Επιτροπή Αξιολόγησης :
Ημερομηνία παρουσίασης:


Υλικό
Αισθητήρας Θερμοκρασίας και Υγρασίας: DC-SS500
Κατασκευαστής: Sure Electronics
Κύκλωμα επεξεργασίας: PIC16F690
Αισθητήρας: HS1101
Πρωτόκολλα σύνδεσης: Αναλογικό, SPI, UART
Mini2440
Υπολογιστής πλακέτα
Κατασκευαστής: FriendlyARM
Επεξεργαστής: 400 MHz Samsung S3C2440A ARM920T (2003)
Μνήμη flash: 128 MB NAND Flash και 2 MB NOR Flash with BIOS (υπάρχουν εκδόσεις με 64MB, 256MB, 1GB NAND Flash επίσης)
Εξωτερική μνήμη: SD-Card


Τι είναι ενσωματωμένα συστήματα
• Φτιαγμένο για μία συγκεκριμένη δουλειά
• Μέρος ενός μεγαλύτερου συστήματος
• Χρησιμοποιούνται κατά κόρον σε ηλεκτρικές και ηλεκτρονικές συσκευές
• Συχνά είναι συστήματα πραγματικού χρόνου
• Περιορισμένες δυνατότητες
• Ιδιαίτερα πρωτόκολλα
• Ιδιαίτερες απαιτήσεις λειτουργικού
• Σημαντική ανάπτυξη και προσδοκίες – smartphones, tablets...


Boot-loader – kernel - openembedded
Boot-loader
• Supervivi
Προεγκατεστημένος, ελλειπής δυνατότητες
• U-boot
Πολύ περισσότερες δυνατότητες, δυνατότητα εκκίνησης από τη NAND
Kernel
• Linux (linux.org)
• Περιέχει Drivers (modules)
• SPI, GPIO
• Busseror, emeb
Openembedded
• Εργαλείο κατασκευής διανομών
• Bitbake (σαν το ebuilds της Gentoo)
• recipes
• Angstrom


Ελεύθερο λογισμικό - λογισμικό ανοιχτού κώδικα Linux
• Ελεύθερο λογισμικό
• Λογισμικό ανοιχτού κώδικα
• GNU – Hurd
• Linux
• Διανομές
open source initiative
FreeBSD
FREEDOM


GPIO - SysFS
SysFS
• System File System (σύστημα αρχείων συστήματος)
• Εικονικό σύστημα αρχείων
• Δεδομένα στη μνήμη
• Mount στο /sys
• Εύκολη πρόσβαση σε συσκευές
GPIO
• General Purpose Input/Output (γενικού σκοπού εισόδου/εξόδου)
• Ακροδέκτες ρυθμιζόμενης λειτουργίας μέσο λογισμικού (κατεύθυνση, τιμή)
• Προσβάσιμα μέσω SysFS στο φάκελο /sys/class/gpio
• export, unexport

LEDS

Θέσεις LED
LED1 --> LED2 gpio38
LED2 --> LED4 gpio40

Export LED 2
```
echo 38 > /sys/class/gpio/export
```

Περιεχόμενα φακέλου:
```
root@mini2440:/sys/devices/virtual/gpio
/gpio38# ls
direction power subsystem uevent value
```

```
root@mini2440:/sys/devices/virtual/gpio/gpio38
# cat direction
in
root@mini2440:/sys/devices/virtual/gpio/gpio38
# cat value
1
```

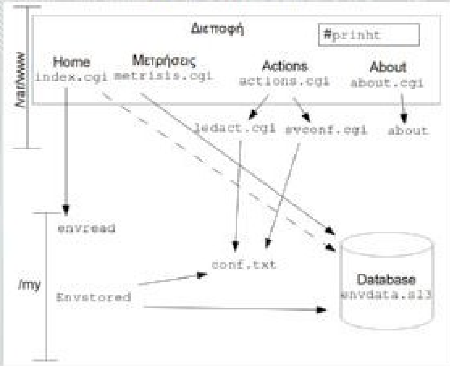


root@mini2440:/sys/devices/virtual/gpio/gpio38# echo out > direction
root@mini2440:/sys/devices/virtual/gpio/gpio38# cat value 0



Κατάργηση:
echo 38 > /sys/class/gpio/unexport

Σχεδιασμός



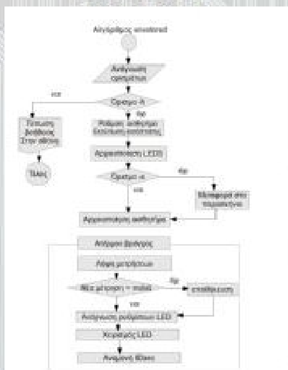
C – SQLite - Webserver

C
- Πολύ διαδεδομένη, εύκολη ανεύρεση πληροφοριών
- Διαθέσιμος cross-compiler για την mini2440
- Δυνατότητα χρήσης σε αρχεία cgi

SQLite
- Ελαφριά
- Επαρκείς δυνατότητες
- Δεν απαιτεί server proses
- Η κάθε βάση δεδομένων είναι ένα αρχείο
- Μπορεί να χρησιμοποιηθεί μέσω της C

Web Server
- Boa
  προεπιλογή, πολύ ελαφρύς, μικρές δυνατότητες
- Cherokee
  ελαφρύς, αρκετές δυνατότητες, δυσκολίες στην εγκατάσταση

envstored



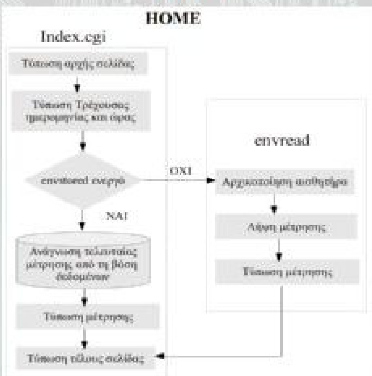
SPI

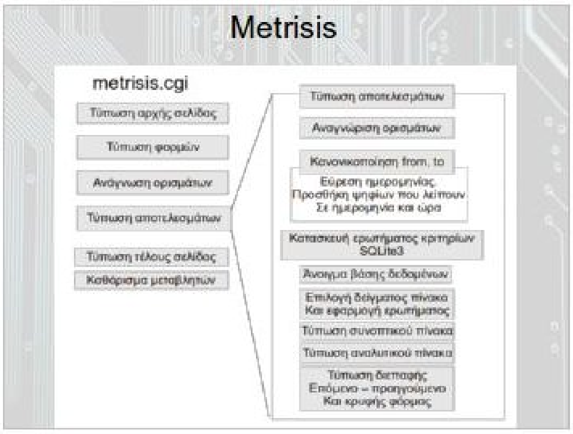
SCLK (Serial Clock): Παλμός ρολογιού που δίνεται από τον αφέντη και χρονίζει την επικοινωνία.
MOSI, SIMO (Master Output, Slave Input): (έξοδος αφέντη, είσοδος σκλάβου) μεταβιβάζει την εντολή από τον αφέντη στο σκλάβο.
MISO, SOMI (Master Input, Slave Output): (είσοδος αφέντη, έξοδος σκλάβου) μεταβιβάζει την απάντηση τού σκλάβου στον αφέντη.
SS (Slave Select): Επιλογή σκλάβου, είναι έξοδος από τον αφέντη και συνήθως ενεργοποιεί το σκλάβο όταν η έξοδος είναι 0.

Mode	CPOL	CPHA
0	0	0
1	0	1
2	1	0
3	1	1

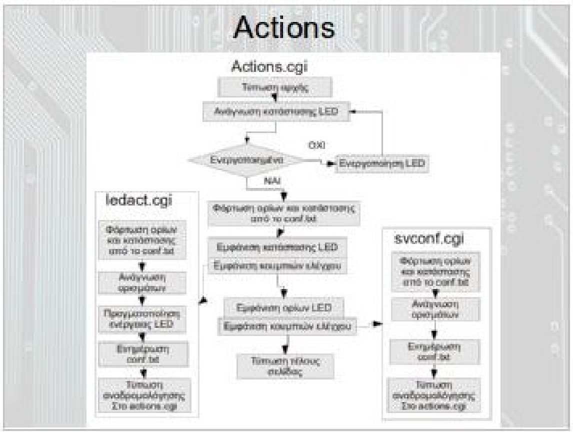
0xa0	Υγρασία σε (%)
0xb0	Θερμοκρασία σε βαθμούς Κελσίου °C
0xb1	Θερμοκρασία σε βαθμούς Φαρενάιτ °F
0xc0	Επιστρέφει 0xAF όταν το σύστημα είναι απασχολημένο.

Home











Περίληψη (εκτενής)

Αυτόνομο σύστημα παρακολούθησης και ελέγχου περιβαλλοντικών συνθηκών βασισμένο στην πλατφόρμα ARM mini2440

Καλαμπούκας Χρήστος *(Author)*
Τμήμα Εφαρμοσμένης Πληροφορικής & Πολυμέσων – ΕΠΠ
Τεχνολογικό Εκπαιδευτικό ίδρυμα Κρήτης
Ηράκλειο, Ελλάδα
line 4: e-mail: xbyte1024@gmail.com

Γεώργιος Κορνάρος *(Επιβλέπων Καθηγητής)*
Τμήμα Εφαρμοσμένης Πληροφορικής & Πολυμέσων – ΕΠΠ
Τεχνολογικό Εκπαιδευτικό ίδρυμα Κρήτης
Ηράκλειο, Ελλάδα
line 4: e-mail: kornaros@epp.teicrete.gr

***Σύνοψη*—Μία αναπτυξιακή πλακέτα με ARM μικροελεγκτή, συνδεδεμένη με ένα αισθητήρα θερμοκρασίας και υγρασίας. Καταγράφει τα δεδομένα του αισθητήρα σε μια βάση δεδομένων και χρησιμοποιεί 2 LED ως ενδείξεις . Παρέχει επίσης ένα web interface που δίνει τη δυνατότητα προβολής των αποτελεσμάτων και χειρισμού των LED.**

Keywords: mini2440, ARM, DC-SS500 , SPI, openembedded, temperature, humidity

I. *Εισαγωγή*

Οι τιμές των ενσωματωμένων συστημάτων έχουν πέσει υπερβολικά, ενώ οι επιδώσεις τους έχουν αυξηθεί τόσο, που θυμίζουν κανονικούς υπολογιστές περασμένης τεχνολογίας. Η διείσδυση τους σε κάθε είδους εφαρμογή είναι τεράστια και τείνουν να αντικαθιστούν λιγότερο ευέλικτες λύσεις, όπως τα, μη προγραμματιζόμενα, ολοκληρωμένα κυκλώματα.

Επιπλέον η εμφάνιση έτοιμων και πολλές φορές ελεύθερων λειτουργικών συστημάτων για αυτά τα συστήματα, κάνει τον προγραμματισμό και την παραμετροποίηση τους ευκολότερη για οποιοδήποτε χρήστη.

Σε αυτή την εργασία βλέπουμε πως μπορεί κάποιος να χρησιμοποιήσει ένα υπολογιστή πλακέτας, εξολοκλήρου με ελεύθερο λογισμικό, και λογισμικό ανοιχτού κώδικα. Καθώς και την ανάπτυξη μιας χρήσιμης εφαρμογής για αυτό το σύστημα.

Η εφαρμογή λαμβάνει μετρήσεις θερμοκρασίας και υγρασίας μέσω ενός αισθητήρα συνδεδεμένου στην πλακέτα και της αποθηκεύει σε μια βάση δεδομένων. Τα δεδομένα είναι προσβάσιμα μέσω μίας διαδικτυακής διεπαφής.

Το σύστημα παρέχει επίσης ανάδραση μέσω δύο LED. Η διεργασία παρασκηνίου ανάβει και σβήνει τα LED ανάλογα με της μετρήσεις που λαμβάνονται, ενώ η διεπαφή δίνει δυνατότητες περαιτέρω ελέγχου σε αυτά.

II. ΥΛΙΚΟ

Το υλικό που χρησιμοποιήθηκε είναι η πλακέτα mini2440 της Friendly ARM που φαίνεται στην (εικόνα 1) . Ο επεξεργαστής που ενεργοποιεί την πλακέτα είναι ο S3C2440 της Samsung, με τον πυρήνα ARM920T και σετ εντολών ARMv4T. Είναι 32bit, RISC επεξεργαστής που πρωτοκυκλοφόρησε το 2003.



εικόνα 1

Ο αισθητήρας είναι ο DC-SS500 της Sure Electronics (εικόνα 2) . Ο κυρίως αισθητήριας που ενσωματώνει είναι ο HS1101 ενώ το κύκλωμα επεξεργασίας PIC16F690 αναλαμβάνει τη μετατροπή των μετρήσεων σε ψηφιακά, κάνοντας δυνατή τη σύνδεση με διάφορα πρωτόκολλα. Το πρωτόκολλο που κρίθηκε πιο βολικό στη συγκεκριμένη περίπτωση είναι το SPI.



εικόνα 2

III. ΛΟΓΙΣΜΙΚΟ

A. Ελεύθερο λογισμικό

Όπως προαναφέρθηκε, για τους σκοπούς της εργασίας έγινε χρίση εξολοκλήρου ελεύθερου λογισμικού, τόσο επάνω στην πλακέτα, όσο και στον κυρίως υπολογιστή. Το ελεύθερο λογισμικό διανέμετε δωρεάν, επομένως κάνει πιο οικονομική την όλη λύση. Επιπλέον το ελεύθερο πνεύμα που πρεσβεύει αρμόζει στην ακαδημαϊκή κοινότητα στην οποία απευθύνεται.
Παλιότερα το ελεύθερο λογισμικό ήταν συχνά ημιτελές και δύσκολο στη χρήση. Τώρα όμως τέτοια προβλήματα είναι πολύ πιο σπάνια, ενώ η τεράστια κοινότητα του τείνει να δίνει λύσεις σε οτιδήποτε.

B. Λογισμικό κυρίως υπολογιστή

Ο υπολογιστής που χρησιμοποιήθηκε για την εγκατάσταση λειτουργικού και προγραμμάτων στην πλακέτα καθώς και για την ανάπτυξη της εφαρμογής τρέχει ελεύθερο λογισμικό. Συγκεκριμένα χρησιμοποιήθηκαν Linux ubuntu ή Mint, openoffice ή Libreoffice, gcc, picocom, gimp, gedit, filezilla.

C. Λειτουργικό πλακέτας

Ως λειτουργικό για την πλακέτα χρησιμοποιήθηκε το openembedded (για συντομία oe). Το openembedded στην πραγματικότητα είναι ένα framework για τη δημιουργία Linux διανομών. Στοχεύει κυρίως στα ενσωματωμένα συστήματα χωρίς να περιορίζεται σε αυτά. Το βασικότερο εργαλείο του είναι το bitbake, το οποίο επιτρέπει να κατεβάσουμε και να χρίσουμε τα προγράμματα που χρειάζονται για το λειτουργικό. Για να το πετύχει αυτό χρησιμοποιεί αρχεία που λέγονται recipes και περιέχουν οδηγίες για το χτίσιμο ενός προγράμματος η μίας ομάδας προγραμμάτων (ακόμα και ολόκληρης της διανομής). Η λειτουργία του θυμίζει το ebuilds της διανομής Gentoo.
Για τη mini2440 καθώς και για πλήθος άλλων συσκευών υπάρχει η διανομή Angstrom. Έχει δημιουργηθεί με τη χρήση του openembedded και περιλαμβάνει βελτιστοποιήσεις για τη συγκεκριμένη συσκευή. Υπάρχει ακόμα και ένας online builder που μπορεί να παράγει κατά παραγγελία για διανομή με βάση τις οδηγίες του χρήστη. Ωστόσο λόγο μικρού ενδιαφέροντος για το συγκεκριμένο υλικό, υπάρχουν κάποια προβλήματα με ορισμένα προγράμματα που μπορεί να επιλεγούν.

D. Πυρήνας

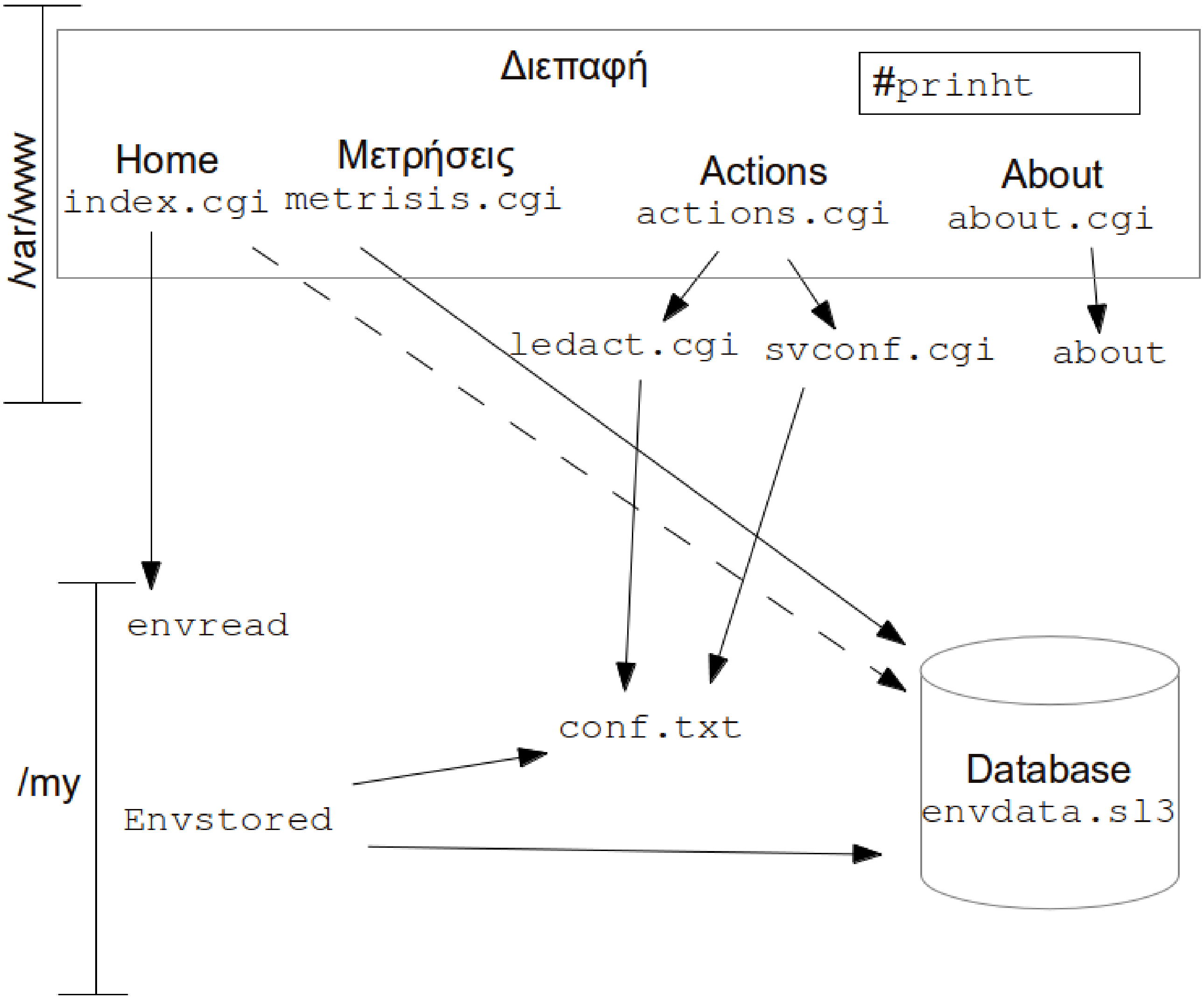
Αν και το openembedded παράγει το δικό του πυρήνα, χρησιμοποιήθηκε ένας άλλος για πιο εύκολη παραμετροποίηση και ενσωμάτωση απαραίτητων οδηγών συσκευών. Κύριο μέλημα αποτέλεσε η υποστήριξη του πρωτοκόλλου SPI το οποίο χρησιμοποιείται για την επικοινωνία πλακέτας – αισθητήρα.

E. Τρόπος εκκίνησης

Αν και η πλακέτα ενσωματώνει μνήμη NAND για την εγκατάσταση του λειτουργικού, επιλέχθηκε η χρήση μίας κάρτας SD για αυτό το σκοπό. Η μνήμη NAND της πλακέτας είναι μόλις 128MB ενώ η αγορά μίας SD κάρτας 2, 4 ή ακόμα και περισσότερων GB είναι αρκετά οικονομική. Με αυτό τον τρόπο έχουμε περισσότερο χώρο για το λειτουργικό και τον πυρήνα ενώ μπορούμε να κρατήσουμε και χώρο swap για εικονική μνήμη. Επιπλέον αν η μνήμη καταστραφεί λόγο εκτεταμένης χρήσης, η αντικατάσταση της SD είναι εύκολη ενώ της μνήμης NAND αδύνατη.
Για να επιτύχουμε αυτό τον τρόπο εκκίνησης χρειάζεται η αντικατάσταση του προεπιλεγμένου boot-loader (supervivi) με κάποιον που να δίνει περισσότερες δυνατότητες. Επιλέγει ο u-boot που επιτρέπει κάτι τέτοιο και δίνει γενικά πολύ περισσότερες δυνατότητες.
Μία από αυτές είναι η δυνατότητα εκκίνησης μέσω δικτύου, όπου η πλακέτα έχει πρόσβαση στο σύστημα αρχείων μέσω NFS. Καθώς τα αρχεία βρίσκονται στον υπολογιστή ελέγχου, είναι πιο εύκολη η πρόσβαση σε αυτά. Ο τρόπος αυτός χρησιμοποιήθηκε σε κάποιο στάδιο ανάπτυξης της εφαρμογής κάνοντας πιο εύκολη της νέας έκδοσης του κάθε αρχείου στο σύστημα.

IV. ΕΦΑΡΜΟΓΗ

Η εφαρμογή που αναπτύχθηκε λαμβάνει και αποθηκεύει μετρήσεις θερμοκρασίας και υγρασίας. Οι μετρήσεις αποθηκεύονται ώστε να είναι προσβάσιμες μέσω της διαδικτυακής διεπαφής. Επιπλέον αν η θερμοκρασία ή η υγρασία υπερβούν κάποιο ανώτατο όριο, ή πέσουν από ένα κατώτατο, ένα LED ανάβει για να μας ειδοποιήσει ανάλογα. Η διεπαφή δίνει δυνατότητα ελέγχου και εποπτείας των LED.
Μπορούμε να χωρίσουμε τα προγράμματα που απαρτίζουν την εφαρμογή σε παρασκήνιο και διεπαφή, ανάλογα με το ρόλο τους και τι θέση τους στο σύστημα αρχείων. Η εικόνα 3 δείχνει όλα τα προγράμματα, τη θέση τους και τη σχέση μεταξύ τους.



εικόνα 3

A. Παρασκήνιο

Το πρόγραμμα που τρέχει στο παρασκήνιο ονομάζεται envstored. Κατά την εκτέλεση του από το τερματικό μπορούμε, με τα κατάλληλα ορίσματα, είτε να το εκτελέσουμε κανονικά (στο παρασκήνιο), είτε να επιλέξουμε να τυπώνετε η έξοδος στην οθόνη (για λόγους αποσφαλμάτωσης κυρίως) είτε να εμφανίσει απλά ένα κείμενο βοήθειας.

Λαμβάνει μετρήσεις θερμοκρασίας και υγρασίας κάθε ένα λεπτό. Αν οι νέες μετρήσεις διαφέρουν από της προηγούμενες, αποθηκεύονται σε μία βάση δεδομένων SQLite3. Έτσι στη βάση δεδομένων έχουμε ένα πίνακα με κάθε αλλαγή των περιβαλλοντικών μεταβλητών.

Επιπλέον με κάθε μέτρηση γίνετε σύγκριση με τις οριακές τιμές των LED τα οποία ανάβουν ή σβήνουν ανάλογα. Τα όρια βρίσκονται αποθηκευμένα στο αρχείο “conf.txt”. Η κατάσταση του LED βέβαια θα μείνει ίδια αν έχουμε επιλέξει χειροκίνητο έλεγχο τον LED από τη διεπαφή.

B. Διεπαφή

Η διεπαφή είναι γραμμένη σε C cgi και τρέχει σε ένα cherokee web server πάνω στην πλακέτα. Τα cgi είναι αρχεία σε κάποια γλώσσα προγραμματισμού (συνήθως Perl), που καλούνται από το server όταν ζητήσουμε την κατάλληλη σελίδα. Κατόπιν εκτελούνται και η έξοδος τους στέλνετε στο web bowser του χρήστη ως κώδικας HTML της σελίδας. Η C επιλέχθηκε γιατί είναι η γλώσσα στη οποία γράφηκε και το πρόγραμμα παρασκηνίου.

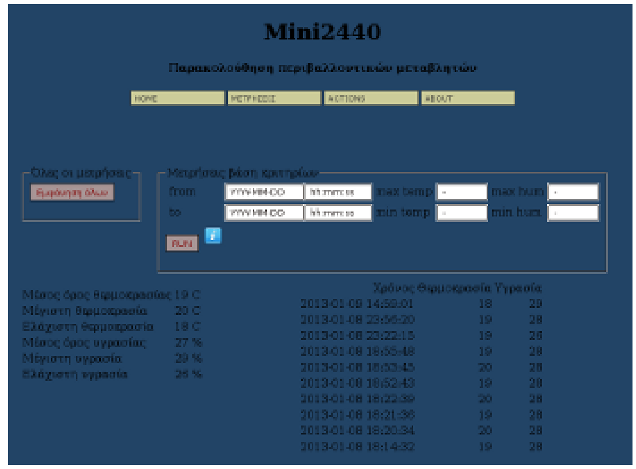
Ο cherokee server επιλέχθηκε λόγο των πολλών δυνατοτήτων και των λίγων πόρων που καταναλώνει. Ωστόσο η έκδοση του στα αποθετήρια της Angstrom διανομής, για την αρχιτεκτονική της mini2440, είναι απαρχαιωμένη και μπορεί να παρουσιάσει προβλήματα ή να λείπουν δυνατότητες.

Η διεπαφή απαρτίζεται από τέσσερα βασικά αρχεία, που το κάθε ένα είναι μια από τις τέσσερις σελίδες της διεπαφής,

και κάποια βοηθητικά αρχεία που καλούνται από το βασικά για να εκτελέσουν κάποιες λειτουργίες. Το αρχείο prinht εκτελείται στην αρχή κάθε βασικού αρχείου για να τυπώσει την αρχή της σελίδας (header), που περιλαμβάνει τον κώδικα css και javascript καθώς και το μενού που βρίσκετε στο πάνω μέρος της κάθε σελίδας. Τα βασικά αρχεία είναι τα:

- index.cgi: Η σελίδα στην οποία οδηγούμαστε από προεπιλογή. Προβάλει την ημερομηνία και ώρα του συστήματος, την τρέχουσα θερμοκρασία και υγρασία, καθώς και η κατάσταση του envstored (ενεργό ή όχι).
  Αν το envstored είναι ενεργό εμφανίζεται η τελευταία μέτρηση από τη βάση δεδομένων. Διαφορετικά καλείτε το envread, το οποίο λαμβάνει τη μέτρηση και τυπώνεται η έξοδος του.
- Metrisis.cgi: Η δεύτερη κατά σειρά σελίδα εμφανίζει της μετρήσεις που βρίσκονται στη βάση δεδομένων. Από προεπιλογή εμφανίζονται η δέκα τελευταίες μετρήσεις. Δίνετε η δυνατότητα εμφάνισης όλων των μετρήσεων ή επιλογής δείγματος με κριτήρια. Τα κριτήρια είναι αρκετά για να περιορίσουμε το δείγμα σε αυτό ακριβός που μας ενδιαφέρει.
  Στο κέντρο της σελίδας εμφανίζονται οι μετρήσεις του δείγματος που επιλέχθηκαν αναλυτικά (στη μορφή ημερομηνία και ώρα – θερμοκρασία – υγρασία). Στα δεξιά εμφανίζονται συνοπτικά στοιχεία για το δείγμα: μέσος όρος, μέγιστο και ελάχιστο, για θερμοκρασία και υγρασία.
- Actions.cgi: Η σελίδα actions εμφανίζει την κατάσταση των LED και δίνει τη δυνατότητα να ελέγξουμε τη συμπεριφορά τους. Για το κάθε LED μπορούμε να επιλέξουμε να μείνει αναμμένο ή σβηστό, ή αυτόματο. Το αυτόματο αφήνει τον έλεγχο του LED στο πρόγραμμα παρασκηνίου που το διαχειρίζεται με βάση τα ανώτερα και κατώτερα όρια. Κάθε αλλαγή στην κατάσταση των LED πραγματοποιείται από το βοηθητικό πρόγραμμα ledact.cgi το οποίο μετά την πραγματοποίηση της ενέργειας επιστρέφει τον έλεγχο στο actions.cgi.
  Τα όρια αυτά εμφανίζονται μέσα στη σελίδα actions από όπου μπορούμε και να τα αλλάξουμε. Οι διαθέσιμες επιλογές είναι μέγιστη θερμοκρασία, μέγιστη υγρασία, ελάχιστη θερμοκρασία και ελάχιστη υγρασία. Όταν μια μέτρηση ξεπεράσει την αντίστοιχη μέγιστη τιμή ανάβει το LED1 ενώ όταν πέσει από την αντίστοιχη ελάχιστη ανάβει το LED2. Οι αλλαγές στα όρια πραγματοποιούνται από το svconf.cgi.
- about.cgi: Η σελίδα about εμφανίζει απλώς κάποιες πληροφορίες για την όλη εργασία και δίνει τη δυνατότητα να τη κατεβάσουμε.

Όπου είναι απαραίτητο, υπάρχουν εικονίδια με το γράμμα "i" που εμφανίζουν πληροφορίες για τη λειτουργία της εφαρμογής. Η εικόνα 4 δείχνει τη σελίδα εμφάνισης μετρήσεων. Είναι είναι ορατό και ένα από τα κουμπιά βοήθειας.



εικόνα 4

V. ΣΥΝΔΕΣΙΜΟΤΗΤΑ

A. GPIO

Το GPIO (General Purpose Input/Output) (γενικού σκοπού εισόδου/εξόδου) είναι ένα σύνολο ακροδεκτών γενικού σκοπού, που η χρήση τους καθορίζεται από το λογισμικό. Πάνω σε αυτούς τους ακροδέκτες μπορεί να συνδεθεί οτιδήποτε.

Ο πιο εύκολος και σύγχρονος τρόπος να ελέγξουμε αυτούς του ακροδέκτες σε Linux είναι το sysfs. Πρόκειται για ένα εικονικό σύστημα αρχείων. Μέσα σε αυτό υπάρχουν αρχεία και φάκελοι που αντιστοιχούν στις υποστηριζόμενες συσκευές. Αλλάζοντας το περιεχόμενο των αρχείων μπορούμε να ενεργοποιήσουμε ή να απενεργοποιήσουμε ακροδέκτες, να καθορίσουμε την κατεύθυνση τους (είσοδος – έξοδος) και την τιμή τους (0 - 1).

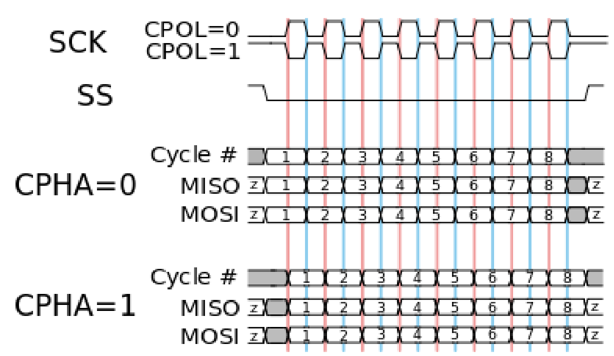
Κάποιοι ακροδέκτες είναι ελεύθεροι, ενώ άλλοι δεσμεύονται από συσκευές της πλακέτας όπως LED, διακόπτες ή και πρωτόκολλα επικοινωνίας.

B. SPI

Η σύνδεση της πλακέτας με τον αισθητήρα έγινε με τη χρήση SPI Serial Peripheral Interface Bus (δίαυλος σειριακής περιφερειακής διεπαφής) . Το οποίο τυποποιήθηκε από τη motorola και είναι ένα πρότυπο μεταφοράς δεδομένων με αρκετές παραλλαγές. Συχνά αναφέρετε ως σειριακός δίαυλος τεσσάρων καλωδίων (Αν και χρησιμοποιείται και έκδοση με τρία καλώδια). Ακολουθεί το μοντέλο αφέντη σκλάβου.

Τα καλώδια που απαιτούνται είναι: παλμός ρολογιού, ενεργοποίηση σκλάβου, δεδομένα αφέντη προς σκλάβο και δεδομένα σκλάβου προς αφέντη.

Η επιλογή λειτουργίας καθορίζετε με τις λειτουργίες SPI (SPI modes). Συνήθως λέγοντας SPI modes αναφερόμαστε στους τέσσερις συνδυασμούς που καθορίζουν πότε στέλνονται και πότε λαμβάνονται μηνύματα. Δύο bit περιγράφουν το mode, το ένα καθορίζει αν η μετάδοση ενεργοποιείται με θετικό ή με αρνητικό παλμό ωρολογίου ενώ το δεύτερο καθορίζει αν η μετάδοση ξεκινάει με την ακμή η την πτώση του παλμού. Η εικόνα 5 βοηθάει να



εικόνα 5

καταλάβουμε αυτή τη λειτουργία.

Άλλες επιλογές είναι το μέγεθος λέξης, η συχνότητα ωρολογίου, η καθυστέρηση μεταξύ δύο εντολών, αν θα σταλεί πρώτα το περισσότερο ή το λιγότερο σημαντικό bit, αν ο σκλάβος ενεργοποιείται στο σήμα 1 ή 0, αν θα ενεργοποιήσουμε τη λειτουργία τριών καλωδίων (μόνο ένα κανάλι δεδομένων).
Υπάρχει επιπλέον η δυνατότητα σύνδεσης πολλών σκλάβων, με ένα καλώδιο ενεργοποίησης για τον κάθε ένα. Συνήθως οι σκλάβοι λειτουργούν σε συγκεκριμένο mode και δε δίνουν τη δυνατότητα αλλαγής του. Ο αφέντης προσαρμόζετε για να επιτευχθεί επικοινωνία. Αν λοιπών η σκλάβοι δε δουλεύουν στο ίδιο mode θα πρέπει ο αφέντης να προσαρμόζετε συνεχώς στης ανάγκες του κάθε ενός, αυξάνοντας την πολυπλοκότητα προγραμματισμού του.
Το SPI έχει πάρα πολλές εφαρμογές. Ανάλογα με την υλοποίηση, τα χαρακτηριστικά του μπορεί να αλλάξουν πολύ. Στην περίπτωση της mini2440 χρησιμοποιούνται τέσσερις από τους ακροδέκτες του GPIO για τις ανάγκες του. Αυτό θέτει κάποιους περιορισμούς στην ταχύτητα, αλλά οι επιδώσεις είναι επαρκείς για τις ανάγκες της εργασίας.
Ο προγραμματισμός του μέσω της C γίνετε χάρη στις βιβλιοθήκες spi.h και spidev.h που περιέχουν τις κατάλληλες συναρτήσεις και παρέχονται από τον κώδικα του πυρήνα του Linux.

VI. ΣΥΜΠΕΡΑΣΜΑΤΑ

Το αποτέλεσμα είναι ένα σύστημα ελέγχου περιβαλλοντικών μεταβλητών, που μπορεί να χρησιμοποιηθεί για τον έλεγχο θερμοκηπίων, αποθηκών και άλλων εγκαταστάσεων που η θερμοκρασία και η υγρασία θεωρούνται σημαντικοί παράγοντες. Επιπλέον αν οι ακροδέκτες των LED συνδεθούν με κατάλληλες συσκευές, όπως συστήματα θέρμανσης, ψύξης ή εξαερισμού, θα μπορούσε αυτόματα να διατηρεί τις επιθυμητές συνθήκες. Με το χειροκίνητο χειρισμό των LED ο έλεγχος αυτός βελτιώνεται. Ενώ χάρη στη διεπαφή που είναι προσβάσιμη μέσο δικτύου είναι εύκολη η απομακρυσμένη πρόσβαση. Επιπλέον υπάρχουν αρκετές δυνατότητες βελτίωσης. Μπορούν να προστεθούν επιπλέον αισθητήρες, είτε για λήψη μετρήσεων από περισσότερα σημεία, είτε για λήψη άλλου τύπου μετρήσεων όπως το PH. Η προσθήκη κάμερας θα μπορούσε να παρέχει και οπτική εποπτεία, ενώ μια οθόνη αφής θα μπορούσε να κάνει το σύστημα πιο αυτόνομο.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] MicroArm Systems , "MINI2440 User's Manual " 2009-03-03

[2] (LinuxMCE wiki) Compil_install_Uboot_Kernel_HowTo.pdf

[3] Sure Electronics Inc "Temperature and Relative Humidity Sensor Module User's Guide " DC-SS500_Ver1.0

[4] Karim Yaghmour , "Building Embedded Linux Systems "

[5] Michael Barr & Antony Massa, "Programming Embedded Systems with C and GNU Developement tools"

[6] Steve,Heath, "Embedded Systems Design"

[7] Bill's Mini2440 Forum "http://billforums.station51.net/index.php"

[8] FriendlyARM official site "http://www.friendlyarm.net/ "

[9] Wikipedia "https://en.wikipedia.org/wiki/Wiki "

[10] Accessing SQLite in C "http://www.linuxjournal.com/content/accessing-sqlite-c "

[11] SQLite official site "http://www.sqlite.org/ "

[12] Installing sqlite headers on ubuntu "http://databasically.com/2010/03/05/installing-sqlite-headers-on-ubuntu-sqlite3-h-not-found/"

[13] How to Daemonize in Linux "http://www-theorie.physik.unizh.ch/~dpotter/howto/daemonize?do=show"

[14] How to create a single instance application in C "http://stackoverflow.com/questions/5339200/how-to-create-a-single-instance-application-in-c"

[15] Documentation:Linux/GPIO "http://www.avrfreaks.net/wiki/index.php/Documentation:Linux/GPIO"

[16] How to Daemonize in Linux "http://www-theorie.physik.unizh.ch/~dpotter/howto/daemonize?do=show"

[17] Footnotes for "CGI Made Really Easy" "http://www.jmarshall.com/easy/cgi/"